

Частное учреждение образования
«Колледж бизнеса и права»

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ «КАЛЬКУЛЯТОР
РАСЧЁТА СТОИМОСТИ АВИАБИЛЕТОВ АВИАКОМПАНИИ
«БЕЛАВИА»
ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по предмету
«Конструирование программ и языки программирования»

КП Т.192010.401

Руководитель проекта

(Е.Н.Коропа)

Обучающийся

(А.А.Носко)

2024

СОДЕРЖАНИЕ

1	Описание задачи	6
1.1	Анализ предметной области	Ошибка! Закладка не определена.
1.2	Постановка задачи	Ошибка! Закладка не определена.
2	Проектирование мобильного приложения	Ошибка! Закладка не определена.
2.1	Проектирование модели	9
2.2	Требования к мобильному приложению	10
2.3	Структура мобильного приложения	Ошибка! Закладка не определена.
2.4	Проектирование макета мобильного приложения	13
2.5	Защита и сохранность данных	21
2.6	Организация и ведение информационной базы (модели)	22
3	Реализация мобильного приложения	25
3.1	Программно-технические средства, необходимые для разработки приложения	Ошибка! Закладка не определена.
3.2	Описание разделов приложения	Ошибка! Закладка не определена.26
3.3	Описание используемых функций и процедур	Ошибка! Закладка не определена.27
3.4	Функциональное тестирование	Ошибка! Закладка не определена.
4	Применение	36
4.1	Назначение программного средства	36
4.2	Условия применения	36
	Заключение	38
	Список использованных источников	39
	Приложение А Текст программных модулей	Ошибка! Закладка не определена.
	Приложение Б Результаты работы программы	Ошибка! Закладка не определена.

					КП Т.192010.401 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Носко А.А.						
Провер.		Коропа Е.Н.						
Т. Контр.								
Н. Контр.								
Утверд.								
						Лит.	Лист	Листов
						у	3	81
						КБП		

Введение

В современном мире путешествия на самолетах становятся все более популярными, и с этим увеличивается потребность в удобных и эффективных инструментах для расчета стоимости авиабилетов. Решение этой задачи вручную может быть сложным и трудоемким, особенно для тех, кто не обладает специальными знаниями и опытом.

Целью курсового проекта является разработка мобильного приложения «Калькулятор расчёта стоимости авиабилетов авиакомпании «Белавиа». Это приложение предназначено для автоматизации процесса расчета стоимости билетов, что поможет пользователям быстро и удобно получать информацию о возможных рейсах и их стоимости.

Актуальность создания такого приложения заключается в его способности обеспечить пользователей актуальной информацией о вылетах, доступными рейсами и их стоимостью на основе различных критериев. Приложение позволит осуществлять поиск билетов по дате вылета, направлению, типу полета (в одну сторону или в обе стороны) и количеству пассажиров. Для каждого рейса будет предоставлена полная информация, включая расчет стоимости перелета с учетом выбранного класса обслуживания, наличия багажа и других параметров.

В рамках курсового проектирования составлена пояснительная записка.

Пояснительная записка состоит из четырех разделов, в которых подробно описаны этапы проектирования и разработки приложения, а также алгоритмы обработки данных и взаимодействия с пользователем.

В первом разделе «Описание задачи» рассматриваются цели создания приложения и проектирование его функционала. Проводится исследование предметной области, анализируются задачи, которые планируется решать с помощью приложения, определяется целевая аудитория, описываются алгоритмы расчета стоимости билетов и обоснование необходимости автоматизации процесса.

Во втором разделе «Проектирование мобильного приложения» описываются средства защиты данных, требования к интерфейсу и функциональным возможностям приложения. Указываются требования к аппаратным и программным ресурсам, включая операционную систему, объем памяти устройства, доступ к интернету и дополнительные устройства.

В третьем разделе «Реализация мобильного приложения» перечисляются инструменты разработки, используемые для создания приложения, порядок авторизации пользователей, организация данных и применяемые технологии.

В четвертом разделе «Применение» рассматривается назначение приложения и анализируются программные средства, необходимые для его функционирования. Описываются возможности и ограничения приложения.

В заключении подводятся итоги выполненной работы, оценивается степень соответствия проектных решений поставленным задачам, указываются возможные несоответствия и причины их возникновения.

В списке информационных источников перечисляются информационные источники, которые были использованы в процессе разработки приложения.

В приложении А представлен текст программных модулей приложения.

В приложении Б представлены результаты работы программы.

Графическая часть содержит диаграммы классов, вариантов использования, последовательности и деятельности.

1 Описание задачи

1.1 Анализ предметной области

Предметной областью решаемой задачи является деятельность авиакомпании «Белавиа» по предоставлению пассажирских авиаперевозок. Объектом решаемой задачи является автоматизация процесса расчета стоимости авиабилетов и управление данными о рейсах.

«Белавиа» является национальной авиакомпанией Беларуси, обеспечивающей широкий спектр внутренних и международных авиаперевозок. Компания базируется в Минске и предлагает свои услуги с высоким уровнем сервиса и безопасности полетов.

Регулярные рейсы: Обеспечивают перевозки пассажиров на основные направления внутри страны и за ее пределами. Включают как короткие внутренние рейсы, так и долгие международные перелеты.

Чартерные рейсы: Предлагают гибкие возможности для организации групповых перевозок, специализированных туров и чартеров для корпоративных клиентов.

Дополнительные сервисы: Включают в себя возможность предварительного выбора места в самолете, услуги по оформлению визы, дополнительные опции питания и комфортные условия для пассажиров с особыми потребностями.

– Рейс – это плановый авиаперелет между двумя аэропортами, включающий в себя вылет, полет по маршруту и приземление, рейс обозначается уникальным номером и может быть выполнен на определенном типе самолета;

– место в самолете – это физическое место, предназначенное для пассажира в кабине самолета. Места различаются по классам обслуживания (эконом, бизнес и первый класс) и могут иметь различные характеристики комфорта;

– Багаж – включает в себя ручной и неручной багаж. ручной багаж пассажиры могут брать с собой в кабину, а неручной багаж передается в багажное отделение самолета;

Авиакомпания «Белавиа» не только обеспечивает надежные и комфортные авиаперевозки, но и стремится к постоянному улучшению своих услуг и удовлетворению потребностей разнообразной аудитории. Современные технологии и высокий уровень обслуживания делают «Белавиа» предпочтительным выбором для путешествий как внутри страны, так и за ее пределами.

– быстрый поиск нужной информации о рейсах. Использование компьютерных технологий позволяет быстро находить информацию о рейсах по заданным критериям;

- расчет стоимости билетов с учетом различных параметров. Это улучшает наглядность и помогает пользователям принимать обоснованные решения.

Потенциальная аудитория пользователей – пассажиры, которые ищут удобные и экономически выгодные рейсы;

- сотрудники авиакомпании, занимающиеся обслуживанием клиентов;
- агентства, занимающиеся продажей авиабилетов и планированием путешествий.

1.2 Постановка задачи

Функции, которые должны быть автоматизированы на основе проанализированных бизнес-процессов и бизнес-задач, представленных ниже.

Парсинг информации о рейсах – программное средство должно позволять автоматически собирать данные о рейсах с официального сайта компании «Белавиа», включая информацию о вылетах, прилетах, направлениях, доступных классах обслуживания и дополнительных услугах.

Поиск рейсов по введенным критериям – пользователи должны иметь возможность вводить параметры поиска, такие как дата вылета и прилета, направление (откуда/куда), тип полета (в одну сторону или в обе стороны), количество пассажиров (взрослые, дети), и получать список доступных рейсов, соответствующих этим критериям.

Отображение полной информации о рейсах – интерфейс приложения должен отображать подробную информацию о каждом найденном рейсе, включая время вылета и прилета, продолжительность полета, доступные классы обслуживания, стоимость билетов и дополнительные параметры (багаж, выбор места, возможность изменений в билете).

Расчет стоимости перелета – приложение должно предоставлять возможность расчета стоимости билета для выбранного рейса с учетом различных параметров, таких как класс обслуживания, количество багажа, выбор места, и другие дополнительные услуги. Результат расчета должен включать детализированную разбивку стоимости по каждому параметру.

Сохранение результатов расчета – пользователи должны иметь возможность сохранять результаты расчета стоимости перелета в файл любого формата (например, PDF, Excel) для дальнейшего использования или отправки по электронной почте.

Удобный и интуитивно понятный интерфейс: интерфейс приложения должен быть простым и интуитивно понятным для любого пользователя, обеспечивая комфортное взаимодействие с программным средством. Это включает создание меню, кнопочных форм и панелей инструментов, которые упрощают навигацию и использование приложения.

Создание справочной системы – приложение должно содержать справочную систему, предоставляющую пользователям необходимую информацию о функциях и возможностях программы, а также руководство по использованию.

Существующие аналоги с указанием отличий, которые будут реализованы в разрабатываемом программном средстве:

- «Skyscanner» – популярное приложение для поиска и бронирования авиабилетов. В отличие от разрабатываемого приложения, «Skyscanner» не позволяет сохранять результаты расчета стоимости билетов в файл и не предоставляет подробной разбивки стоимости по параметрам;

- «Momondo» – еще одно известное приложение для поиска авиабилетов. Основное отличие заключается в том, что разрабатываемое приложение будет иметь более простой и интуитивно понятный интерфейс, а также функции автоматического парсинга данных с официального сайта авиакомпании; [6]

- «Kayak» – мощное приложение для поиска и бронирования билетов с широкими возможностями фильтрации и анализа. Однако, оно не позволяет сохранять результаты расчета стоимости в файл и не предоставляет детализированную разбивку стоимости по параметрам;

Разрабатываемое программное средство будет отличаться от существующих аналогов более удобным интерфейсом, автоматизацией парсинга данных с сайта авиакомпании, возможностью детализированного расчета стоимости билетов и сохранения результатов в файл, что обеспечит пользователям более комфортное и эффективное взаимодействие с приложением.

2 Проектирование мобильного приложения

2.1 Проектирование модели

Проектирование модели мобильного приложения включает в себя создание диаграмм, которые являются ключевыми элементами в документации и помогают визуализировать структуру и функциональность приложения.

Диаграмма последовательности (Sequence Diagram) иллюстрирует взаимодействие между объектами в определенной последовательности времени, показывая, как сообщения передаются между объектами в рамках выполнения конкретного сценария. В контексте разработки мобильного приложения «Калькулятор расчёта стоимости авиабилетов авиакомпании «Белавиа», диаграмма последовательности будет отображать основные этапы выполнения прецедентов, таких как:

- поиск рейсов;
- расчет стоимости билета;
- сохранение результатов;
- просмотр информации о рейсах;
- просмотр справочной системы;

Данная диаграмма представлена в графической части на листе 3.

Диаграмма деятельности (Activity Diagram) иллюстрирует последовательность действий или процессов в системе. Она подходит для визуализации более сложных процессов и позволяет понять поток выполнения операций в приложении, данная диаграмма представлена в графической части на листе 4.

Диаграмма вариантов использования (use case diagram) иллюстрирует взаимодействие пользователя с системой, показывая, какие функции доступны и как они связаны между собой. В этой диаграмме выделены основные прецеденты, такие как:

- поиск рейсов;
- расчет стоимости билета;
- сохранение результатов;
- просмотр информации о рейсах;
- просмотр справочной системы.

Для каждого из этих вариантов использования указаны типы связей, например, ассоциации, расширения и включения.

Поиск рейсов – пользователь вводит параметры поиска (дата, направление, количество пассажиров) и запускает процесс поиска.

Расчет стоимости билета – пользователь выбирает рейс из предложенного списка и вводит дополнительные параметры (класс обслуживания, багаж, выбор места). Приложение выполняет расчет стоимости. [4]

Сохранение результатов – пользователь сохраняет результаты расчета в файл.

Просмотр информации о рейсах – пользователь может просматривать детализированную информацию о каждом рейсе.

Просмотр справочной системы – пользователь может просматривать справочную информацию о приложении

Данная диаграмма представлена в графической части на листе 1.

Диаграмма классов – иллюстрирует модель данных и взаимосвязи между классами в приложении. «Классы могут включать «Рейс», «Билет» и «Пользователь», с атрибутами и методами, необходимыми для функционирования приложения, такими как цена, дата, методы поиска и расчёта стоимости.

Данная диаграмма представлена в Графической части 2

Структура мобильного приложения – отображает архитектуру приложения, включая слои пользовательского интерфейса, бизнес-логики и доступа к данным. Данная диаграмма представлена в Графической части 2.

2.2 Требования к мобильному приложению

Мобильное приложение «Калькулятор расчёта стоимости авиабилетов авиакомпании «Белавиа»» должно поддерживать следующие операционные системы:

- android – версия 6.0 (Marshmallow) и выше. Это позволит охватить большую часть пользователей Android-устройств;

- iOS – версия 12.0 и выше. Это обеспечит совместимость с большинством современных устройств Apple.

Приложение должно иметь единый стиль оформления, который соответствует корпоративной идентичности авиакомпании «Белавиа». Основные требования включают:

- использование фирменных цветов компании для элементов интерфейса (логотип, кнопки, панели);

- единый стиль для всех экранов и элементов управления, включая иконки и кнопки;

- современный и чистый дизайн, обеспечивающий легкость восприятия и удобство использования.

Графический дизайн приложения должен быть интуитивно понятным и эстетически приятным. Основные требования:

- высокое качество графических элементов, таких как иконки, кнопки и изображения;

- адаптивный дизайн, который корректно отображается на устройствах с разными разрешениями экрана;

– минималистичный подход, чтобы избежать перегруженности интерфейса и обеспечить легкую навигацию.

Шрифтовое оформление должно обеспечивать читаемость и соответствовать общему стилю приложения. Основные требования:

– использование стандартных шрифтов, поддерживаемых как Android, так и iOS (например, Roboto для Android и San Francisco для iOS);

– размер шрифта должен быть достаточно крупным для комфортного чтения на экранах различных размеров;

– четкий контраст между текстом и фоном для улучшения читаемости.

Приложение должно быть оптимизировано для работы на различных типах мобильных устройств:

– смартфоны с диагональю экрана от 4 до 7 дюймов;

– планшеты с диагональю экрана от 7 до 10 дюймов;

– минимальные требования к памяти устройства 2 ГБ оперативной памяти и 100 МБ свободного пространства для установки приложения;

– доступ к интернету для парсинга данных и обновления информации о рейсах.

Контент приложения должен быть актуальным, точным и полезным для пользователей. Основные требования:

– полное и детализированное описание каждого рейса, включая информацию о времени вылета и прилета, классе обслуживания, наличии багажа и возможности изменений в билете.

Компоновка экранов должна быть логичной и удобной для пользователя. Основные требования:

– главный экран с основными функциями и возможностями поиска рейсов;

– экран поиска с полями для ввода критериев поиска (дата, направление, количество пассажиров);

– экран результатов поиска с подробной информацией о найденных рейсах;

– экран расчета стоимости с возможностью выбора дополнительных параметров и сохранения результатов;

– экран настроек с возможностью изменения языка, настроек уведомлений и других параметров;

– справочная система, доступная с любого экрана приложения для быстрого доступа к информации о функциях приложения.

Эти требования обеспечат создание удобного, функционального и привлекательного мобильного приложения, соответствующего ожиданиям пользователей и стандартам авиакомпании «Белавиа».

2.3 Структура мобильного приложения

Логическая структура мобильного приложения «Калькулятор расчёта стоимости авиабилетов авиакомпании «Белавиа»[1] представлена на диаграмме компонентов, где показаны основные модули и их взаимодействие. Основные модули включают:

- user Interface (UI) – обеспечивает взаимодействие с пользователем, включает в себя экраны и элементы управления;
- business Logic (BL) – реализует основную функциональность приложения, такую как поиск рейсов, расчет стоимости и управление данными;
- data Access Layer (DAL) – отвечает за доступ к данным, включая парсинг информации с сайта авиакомпании и работу с локальными базами данных;
- network Module – обеспечивает взаимодействие с интернетом для получения актуальной информации о рейсах и их стоимости.

Физическая структура проекта на языке C# с использованием Xamarin организована следующим образом:

```
/BelaviaTicketCalculator
|— /BelaviaTicketCalculator.Android
|   |— Resources
|   |   |— drawable
|   |   |— layout
|   |   |— values
|— /BelaviaTicketCalculator.iOS
|   |— Resources
|   |   |— Images.xcassets
|— /BelaviaTicketCalculator
|   |— Models
|   |— ViewModels
|   |— Views
|   |— Services
|   |— Helpers
|   |— App.xaml
|   |— MainPage.xaml
|   |— MainPage.xaml.cs
|   |— FavoriteFlightsPage.xaml
|   |— FavoriteFlightsPage.xaml.cs
|   |— FlightDetailsPage.xaml
|   |— FlightDetailsPage.xaml.cs
|   |— Login.xaml
|   |— Login.xaml.cs
|   |— MainPag.xaml
|   |— MainPag.xaml.cs
```

| |— Page1.xaml
| |— Page1.xaml.cs
/BelaviaTicketCalculator.Android и /BelaviaTicketCalculator.iOS – платформозависимые проекты, содержащие ресурсы и специфические для платформы файлы.
/BelaviaTicketCalculator/Models – содержит классы моделей данных.
/BelaviaTicketCalculator/ViewModels – содержит классы моделей представлений, реализующие логику приложения.
/BelaviaTicketCalculator/Views – содержит XAML-файлы макетов экранов.
/BelaviaTicketCalculator/Services – содержит классы для работы с данными и сетевыми взаимодействиями.
/BelaviaTicketCalculator/Helpers – содержит вспомогательные классы и утилиты.
App.xaml и App.xaml.cs – файл приложения, содержащий глобальные ресурсы и инициализацию.

Структура каждого макета экрана должна быть организована таким образом, чтобы обеспечить интуитивно понятный и эффективный пользовательский опыт. Например, экран поиска рейсов может включать поля для ввода даты вылета и прилета, направления, и выбора класса обслуживания.

2.4 Проектирование макета мобильного приложения

Графические макеты и прототипы экранов были созданы с использованием онлайн – сервиса как Figma. Эти макеты служат визуальным представлением интерфейса приложения и помогают в процессе разработки и тестирования.

На рисунке 2.4.1 представлен прототип «Главное окно»

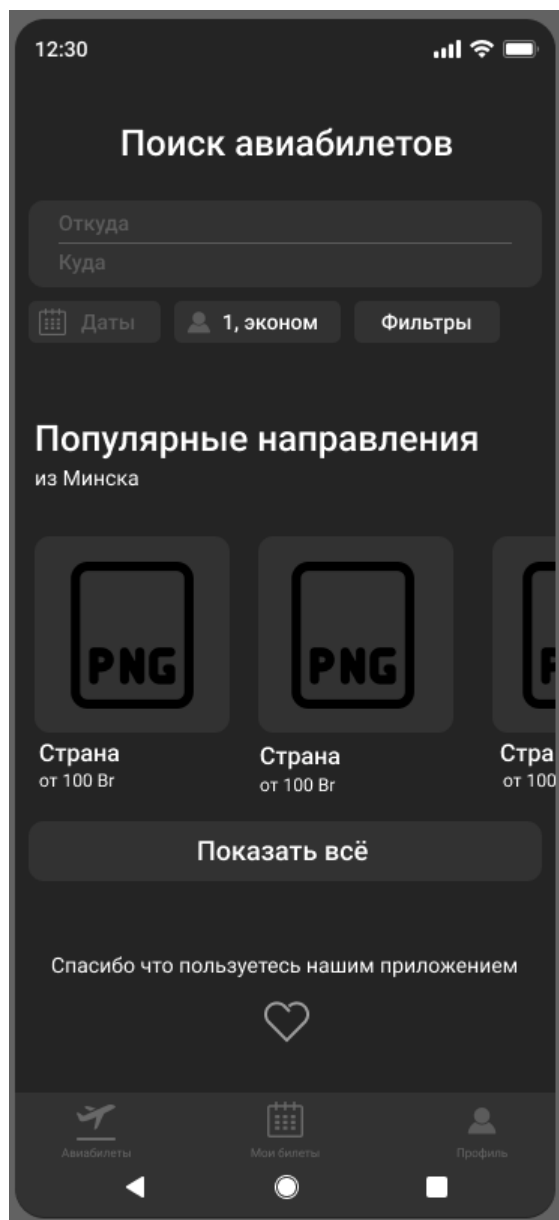


Рисунок 2.4.1 – Макет «Главное окно»

Содержимое главного окна включает следующие:

- текстовое поле «Откуда» позволяет ввести страну, откуда будет искаться авиаперелёт;
- текстовое поле «Куда» позволяет ввести страну, куда будет искаться авиаперелёт;
- кнопка «Даты» позволяет выбрать дату авиаперелёта;
- кнопка «1, эконом» позволяет выбрать количество пассажиров и класс перелёта;
- кнопка «Фильтры» позволяет выбрать перелёт с багажом или без;
- на главном экране показаны популярные направления; кнопка «Показать всё» позволяет открыть все популярные направления из Минска;
- в панели навигации кнопка «Мои билеты» позволяет перейти на вкладку сохранённых билетов;

– кнопка «Профиль» позволяет перейти в профиль или зарегистрировать его.

На рисунке 2.4.2 представлен прототип «Мои Билеты»

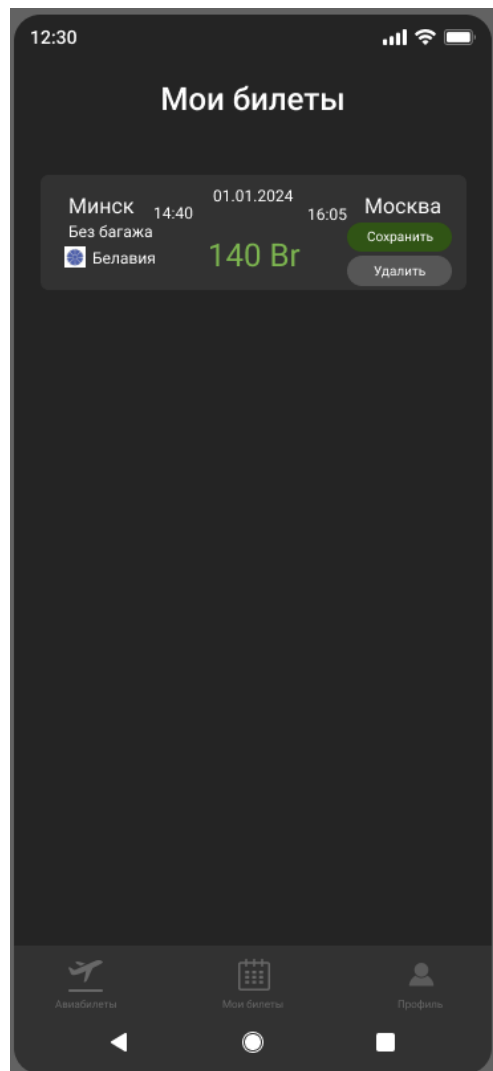


Рисунок 2.4.2 – Макет «Мои Билеты»

Содержимое окна «Мои билеты» включает следующие:

– билет, который можно сохранить или удалить, а также подробная информация о нём;

– панель навигации для удобства использования приложения.

На рисунке 2.4.3 представлен прототип «Поиск билета»

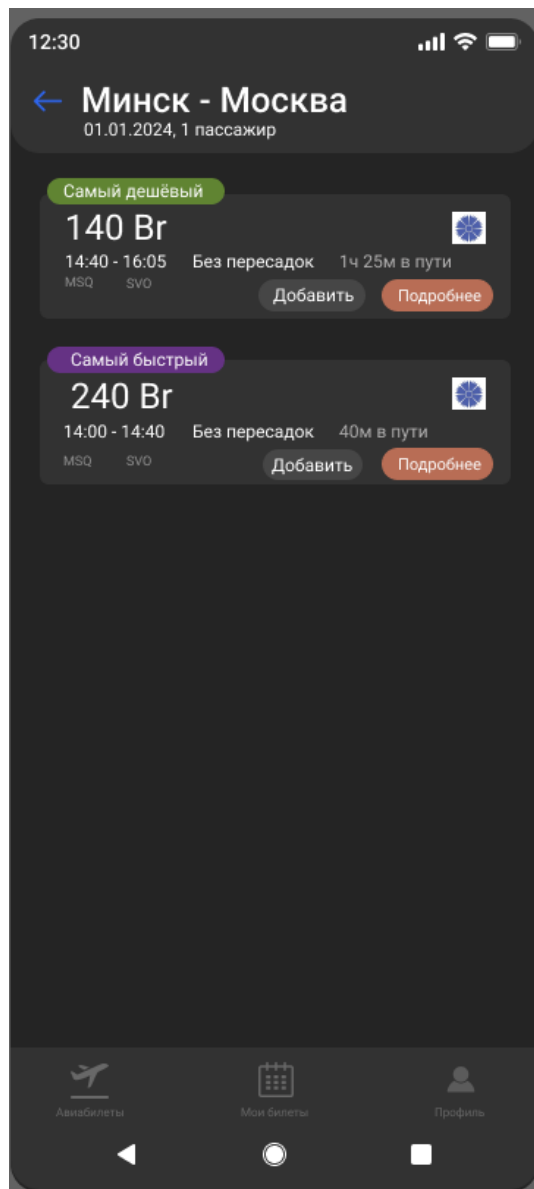


Рисунок 2.4.3 – Макет «Поиск билета»

Содержимое окна «Поиск билета» включает следующие:

- сохранить в разделе «Мои билеты»;
- подробнее – нажав на эту кнопку, откроется полная информация о билете, включая расчёт стоимости;
- панель навигации обеспечивает лёгкий доступ ко всем функциям приложения.

На рисунке 2.4.4 представлен прототип «Выбор даты»

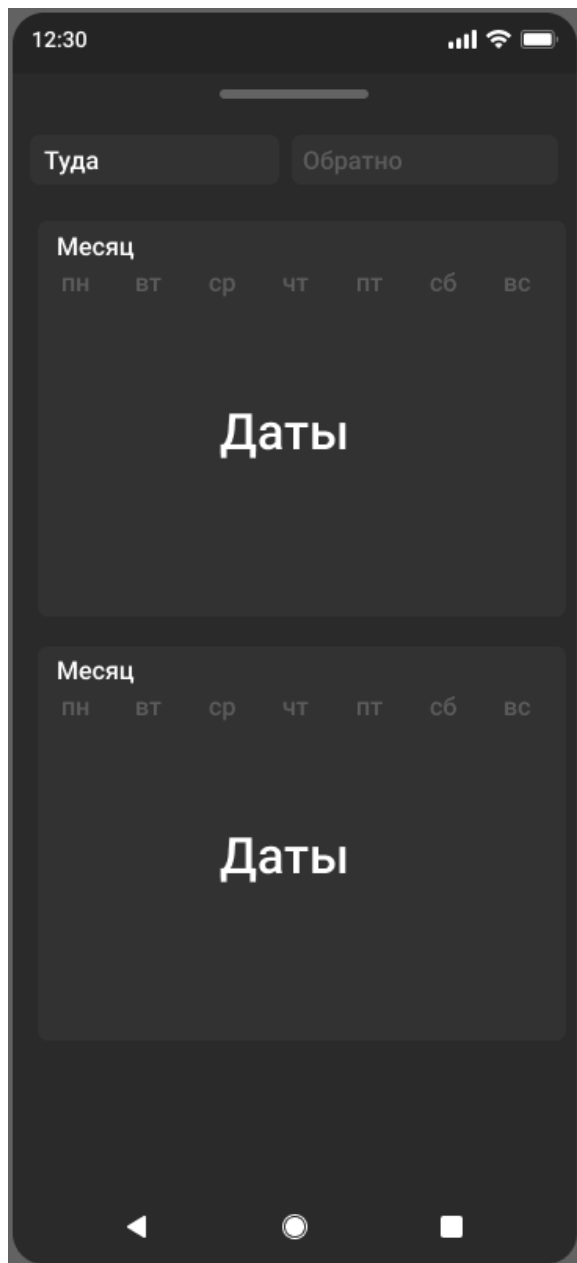


Рисунок 2.4.4 – Макет «Выбор даты»

Содержимое окна «Выбор даты» включает следующие:

- на экране отображены два календаря первый указывает дату вылета, второй – дату обратного полёта;
- сверху расположен слайдер, который позволяет закрыть окно с датами.

На рисунке 2.4.5 представлен прототип «Профиль»

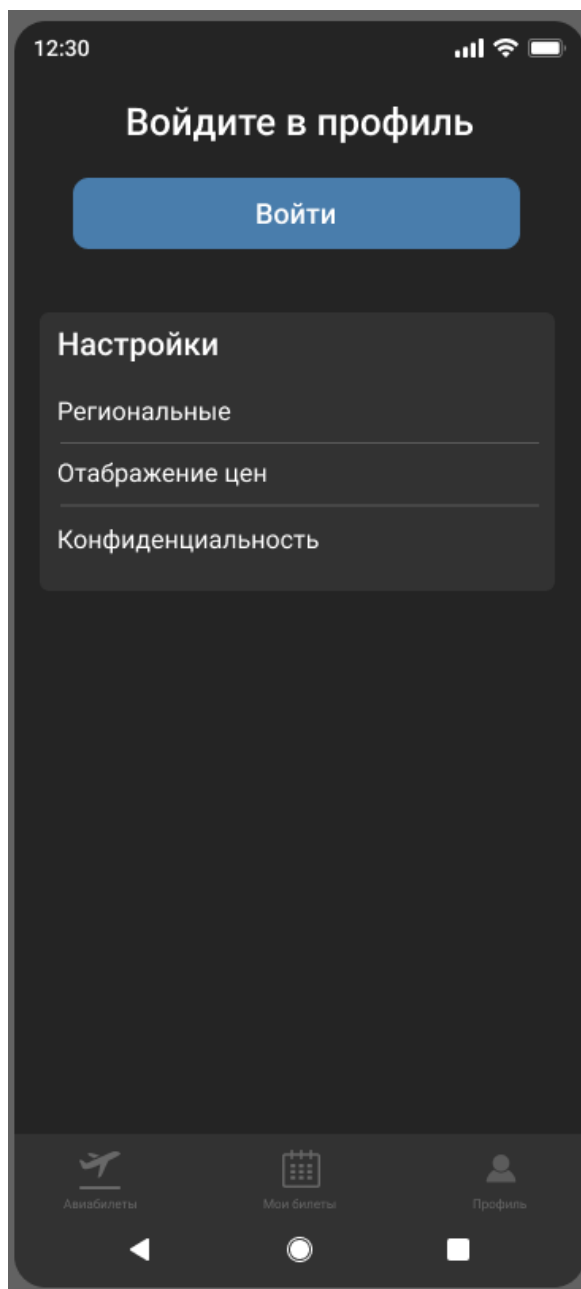


Рисунок 2.4.5 – Макет «Профиль»

Содержимое окна «Профиль» включает следующие:

- кнопка «Войти» позволяет пользователю авторизоваться или зарегистрироваться в приложении;
- кнопка «Региональные настройки» дает возможность выбрать локализацию;
- кнопка «Отображение цен» позволяет выбрать формат, в котором будут показаны цены;
- кнопка «Конфиденциальность» предоставляет доступ к настройкам приватности пользователя.

На рисунке 2.4.6 представлен прототип «Параметры поиска»

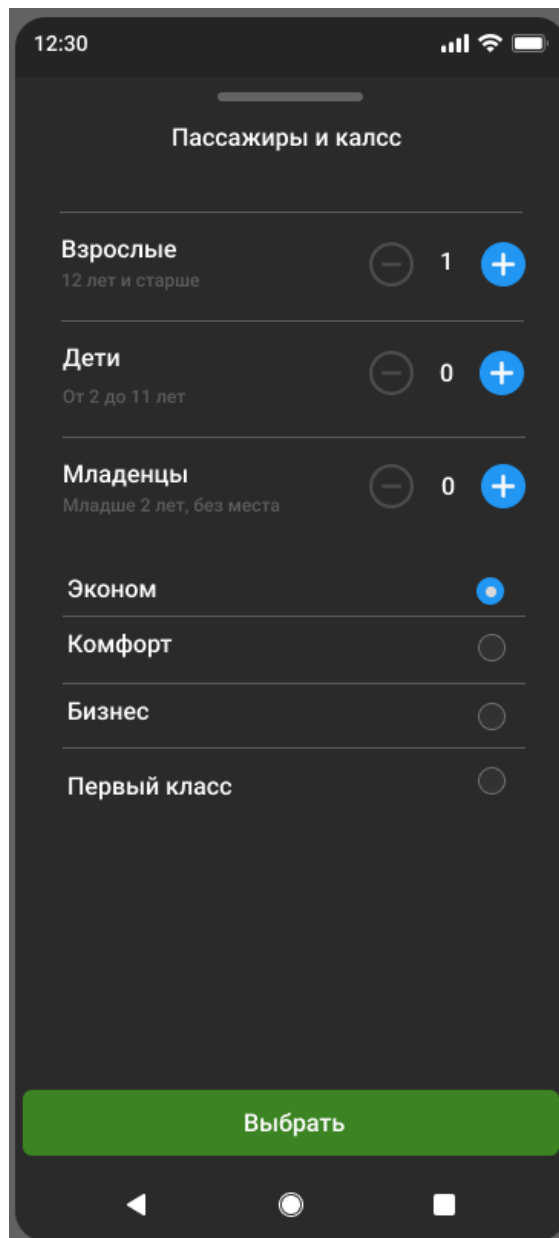


Рисунок 2.4.6 – Макет «Параметры поиска»

Содержимое окна «Параметры поиска» включает следующие:

- кнопки выбора пассажиров для указания количества путешествующих;
- кнопка выбора класса позволяет определить уровень комфорта полёта;
- главная кнопка «Выбрать» для сохранения настроек параметров поиска.

На рисунке 2.4.7 представлен прототип «Выбор страны»

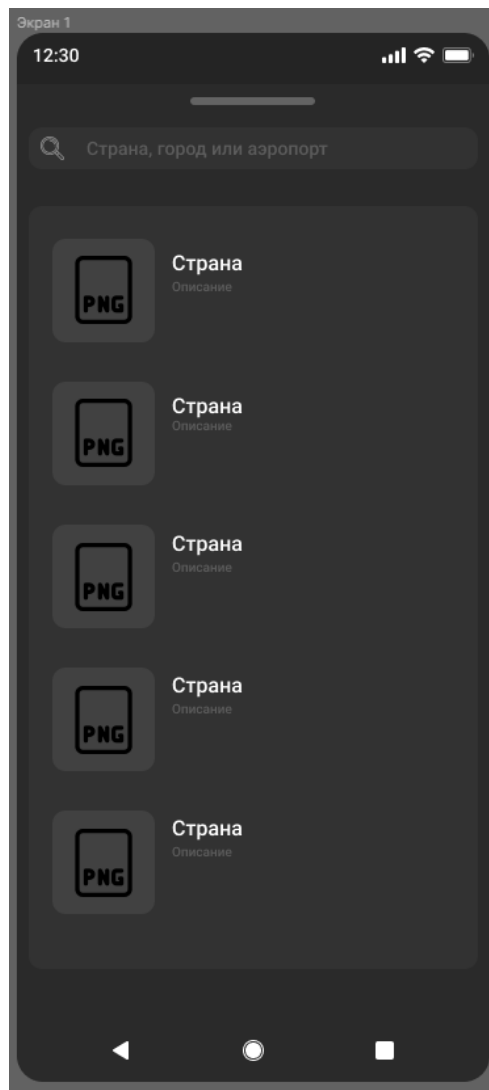


Рисунок 2.4.7 – Макет «Выбор страны»

Содержимое окна «Выбор страны» включает следующие:

- на экране отображено множество стран, из которых можно выбрать пункт отправления или назначения для поиска авиарейса.

На рисунке 2.4.8 представлен прототип «Фильтр билета»

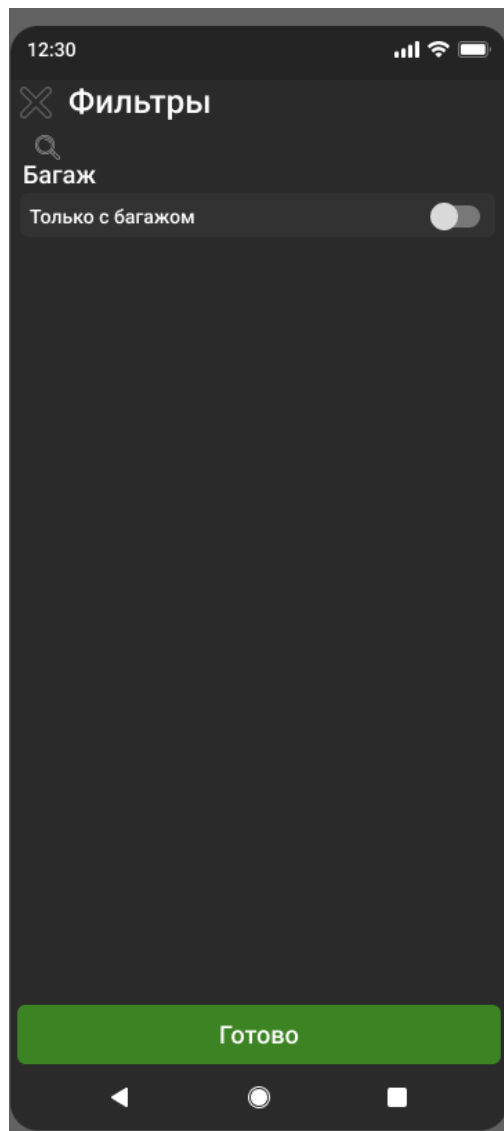


Рисунок 2.4.8 – Макет «Фильтр билета»

Содержимое окна «Фильтр билета» включает следующие:

- кнопка «Только с багажом» позволяет искать авиабилеты, которые включают багаж;
- кнопка «Готово» сохраняет выбранные параметры поиска.

На рисунке 2.4.9 представлен прототип «Окно входа»

2.5 Защита и сохранность данных

Ограничение доступа к данным может быть реализовано через систему аутентификации и авторизации, где пользователи должны войти в систему, чтобы получить доступ к определенным функциям. Разрешения (permissions)

определяют, какие действия пользователь может выполнять после входа в систему. Защита информации от несанкционированного использования включает в себя шифрование данных, безопасное хранение паролей и использование безопасных протоколов передачи данных. Для обеспечения безопасности будет использоваться хеш-шифрование.

2.6 Организация и ведение информационной базы (модели)

Информационная база приложения «Калькулятор расчёта стоимости авиабилетов авиакомпании «Белавиа»» состоит из различных таблиц/файлов, отражающих содержание информационных сущностей, таких как пользователи, рейсы, билеты и другие. Каждая таблица имеет описание своих полей и связей с другими таблицами, что обеспечивает целостность и логичность структуры базы данных.

Таблица «Пользователь» включает следующие данные:

- Id с типом int – уникальный идентификатор пользователя;
- email с типом string – электронная почта пользователя;
- пароль с типом string – пароль для доступа к аккаунту;

Таблица «Рейс» включает следующие данные:

- id с типом int – уникальный идентификатор рейса;
- номер_рейса с типом string – номер авиарейса;
- город_отправления_id с типом int – идентификатор города отправления;

- город_прибытия_id с типом int – идентификатор города прибытия;

- время_вылета с типом datetime – время вылета;
- время_прилета с типом datetime – время прилета;

Таблица «Город» включает следующие данные:

- id с типом int – уникальный идентификатор города;
- название с типом string – название города;
- страна с типом string – страна, в которой находится город;

Таблица «Авиакомпания» включает следующие данные:

- Id с типом int – уникальный идентификатор авиакомпании;
- название с типом string – название авиакомпании;
- логотип с типом string – путь к логотипу авиакомпании;

Таблица «Аэропорт» включает следующие данные:

- Id с типом int – уникальный идентификатор аэропорта;
- название с типом string – название аэропорта;
- город_id с типом int – идентификатор города, в котором находится аэропорт;

Таблица «Класс» включает следующие данные:

- id с типом int – уникальный идентификатор класса обслуживания;
- название с типом string – название класса (эконом, бизнес);

Таблица «Билет» включает следующие данные:

- id с типом int – уникальный идентификатор билета;
- рейс_id с типом int – идентификатор рейса;
- пассажир_id с типом int – идентификатор пассажира;
- класс_id с типом int – идентификатор класса обслуживания;
- цена с типом decimal – стоимость билета;

Таблица «Пассажир» включает следующие данные:

- Id с типом int – уникальный идентификатор пассажира;
- имя с типом string – имя пассажира;
- фамилия с типом string – фамилия пассажира;
- паспорт с типом string – номер паспорта;

Взаимосвязи таблиц между собой выполнены следующим образом:

- пользователь и Билет – отношение «один ко многим» – один пользователь может иметь множество билетов;
- рейс и Билет – отношение «один ко многим» – один рейс может быть связан с множеством билетов;
- город и Аэропорт – отношение «один ко многим» – в одном городе может быть несколько аэропортов;
- авиакомпания и Рейс – отношение «один ко многим» – одна авиакомпания может выполнять множество рейсов;
- класс и Билет – отношение «один ко многим» – один класс обслуживания может быть применён к множеству билетов;
- пассажир и Билет – отношение «один к одному» – каждый билет привязан к одному пассажиру.

Схема модели базы данных представляет собой визуальное представление взаимосвязей между таблицами и их полями представлена на рисунке 2.6.1

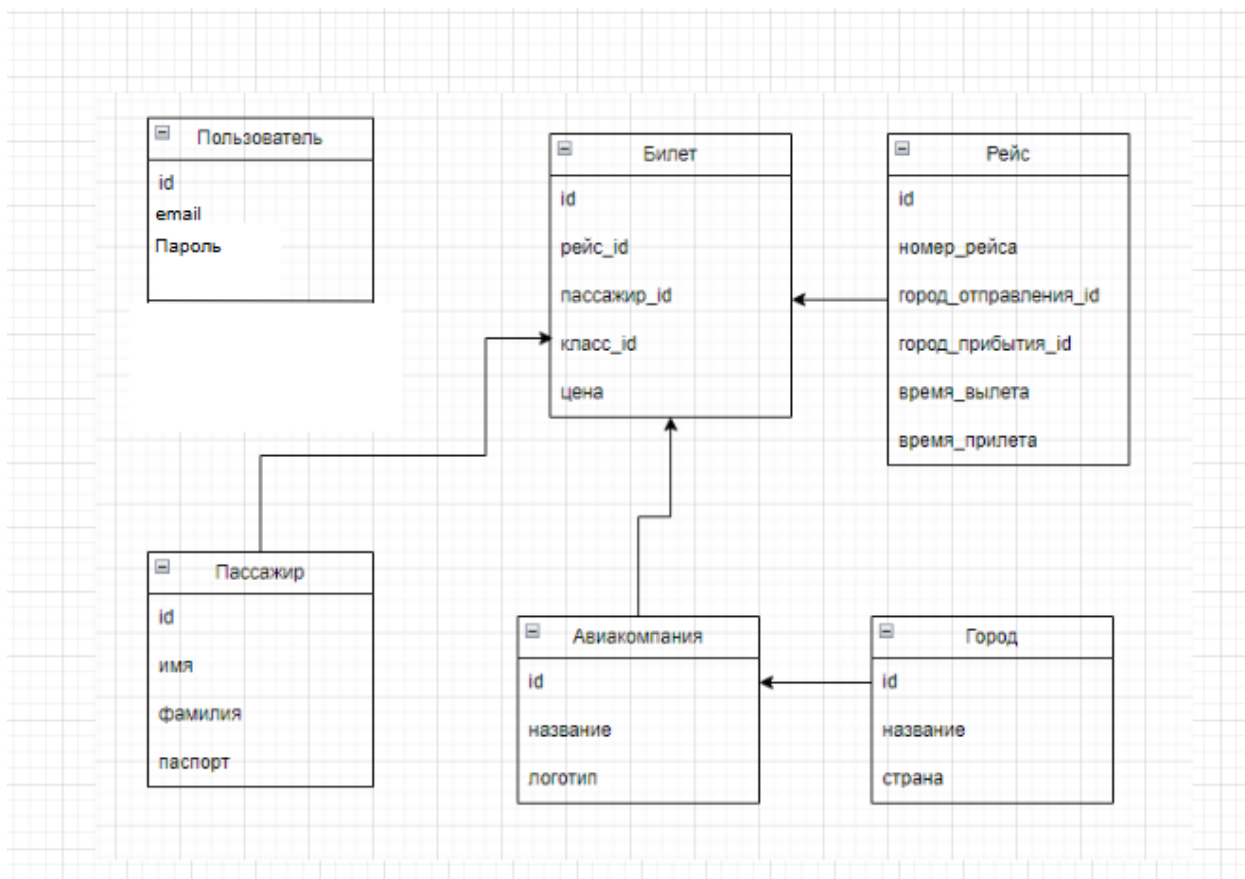


Рисунок 2.6.1 – Схема модели базы данных

3 Реализация мобильного приложения

3.1 Программно–технические средства, необходимые для разработки приложения

При разработке мобильного приложения «Калькулятор расчёта стоимости авиабилетов авиакомпании «Белавиа»» важно правильно выбрать инструменты и технологии, которые будут использованы. Основные критерии выбора включают удобство разработки, поддерживаемые платформы, производительность, наличие необходимого функционала и возможность дальнейшего масштабирования.[2]

Обоснование выбранных инструментов разработки приложения

Для разработки данного мобильного приложения был выбран язык программирования C# и платформа Xamarin.

C# – Язык программирования C# является мощным и гибким языком, который поддерживает объектно–ориентированное программирование и имеет богатую стандартную библиотеку. Он используется для разработки приложений на различных платформах, включая Windows, Android и iOS.[9]

Xamarin – Xamarin представляет собой платформу для кроссплатформенной разработки мобильных приложений. Она позволяет разрабатывать приложения для Android и iOS с использованием единого базового кода на C#. Преимущества Xamarin включают:

- возможность использования одного кода для разных платформ, что значительно сокращает время разработки и облегчает поддержку приложения;
- поддержка использования родных API и библиотек, что позволяет создавать высокопроизводительные приложения с родным интерфейсом;
- интеграция с Visual Studio, что обеспечивает удобство разработки и отладки.

Для реализации мобильного приложения использованы следующие технологии:

- C# с .NET ; [9];
- Xamarin;

.NET – Это кроссплатформенный фреймворк, который предоставляет широкий набор инструментов и библиотек для разработки приложений. Он обеспечивает высокую производительность и безопасность, а также поддерживает работу с различными базами данных и веб–сервисами.

Xamarin.Forms – является частью платформы Xamarin, предназначенной для создания интерфейсов пользователей (UI), которые могут быть использованы на нескольких платформах. С его помощью можно создавать UI, которые адаптируются под особенности каждой платформы, обеспечивая при этом единый код.

Xamarin.Android и Xamarin.iOS – Эти компоненты позволяют использовать родные API Android и iOS, что обеспечивает доступ к функционалу устройств на уровне родных приложений. [9]

Visual Studio – Основная среда разработки, предоставляющая все необходимые инструменты для написания, отладки и тестирования кода. [10] Visual Studio имеет интеграцию с Xamarin, что позволяет эффективно управлять проектом и взаимодействовать с устройствами Android и iOS.

Единая кодовая база – Использование Xamarin позволяет значительно сократить время разработки и упростить поддержку приложения благодаря возможности использования единого кода для нескольких платформ.

Высокая производительность – Xamarin предоставляет доступ к родным API, что позволяет создавать высокопроизводительные приложения с родным интерфейсом.

Удобство разработки – Интеграция с Visual Studio и использование C# обеспечивают высокое удобство разработки и отладки, что снижает вероятность возникновения ошибок и ускоряет процесс разработки.

Таким образом, выбор C# с .NET и Xamarin для разработки мобильного приложения «Калькулятор расчёта стоимости авиабилетов авиакомпании «Белавиа»» является обоснованным и эффективным решением, обеспечивающим высокое качество и производительность приложения.

3.2 Описание разделов приложения

Главный экран приложения «Калькулятор расчёта стоимости авиабилетов авиакомпании «Белавиа»» является центральной точкой входа для пользователя и предоставляет основные функции приложения. Он включает следующие элементы:

- заголовок отображает заголовок с надписью «Поиск авиабилетов»;
- поле поиска рейсов отображать форму, которая позволяет пользователю ввести критерии поиска билетов.
- дата вылета – выбор даты вылета рейса;
- дата прилета – выбор даты возвращения;
- направление – поля для ввода города отправления и города прибытия;
- тип поездки – выбор класса и количества пассажиров;
- фильтр – ввод поиска авиабилета с багажом;
- список результатов – Отображает результаты поиска рейсов с возможностью выбора конкретного рейса для детальной информации и расчёта стоимости;

Приложение также включает навигационное меню, которое обеспечивает доступ к различным разделам и функциям приложения. Меню включает следующие пункты:

- авиабилеты – возвращает пользователя на главный экран приложения, где пользователь может выполнить новый поиск рейсов;
- мои билеты – переход на экран, где пользователь сохранял билеты и может их просмотреть ещё раз;

– профиль – переходит на экран, где пользователь может настроить свой профиль;

Каждый пункт меню выполняет конкретное действие, облегчающее навигацию по приложению и предоставляющее доступ к необходимым функциям и информации.

3.3 Описание используемых функций и процедур

Система для хранения данных: Спроектировать и реализовать систему для хранения информации о рейсах, билетах, параметрах расчета и других необходимых данных.

Парсинг информации: Осуществить парсинг информации о вылетах с официального сайта авиакомпании для актуализации данных в приложении.

Функции поиска билетов: Реализовать функцию поиска билетов на рейсы по следующим критериям:

- Дата вылета (или прилета)
- Направление (откуда/куда)
- Тип перелета (в одну сторону, туда и обратно)
- Пассажиры (количество, возрастные категории: взрослые, дети)

Вывод информации о рейсах: Для выбранного списка вылетов предоставить полную информацию о всех доступных рейсах, включая детали о времени вылета и прилета, типе самолета и доступных классах обслуживания.

Расчет стоимости перелета: Для выбранного пользователем рейса из предложенного списка осуществить расчет стоимости перелета с учетом выбранных параметров:

- Класс обслуживания (эконом, бизнес и т.д.)
- Выбор места
- Параметры багажа
- Возможность изменений в билете

Сохранение расчета – предусмотреть возможность сохранения расчета стоимости перелета в файле любого формата для последующего использования или архивации.

Организация интерфейса – обеспечить лаконичный и интуитивно понятный интерфейс приложения с использованием элементов управления, таких как меню, кнопочные формы и панели инструментов.

Иерархия классов – создать соответствующую иерархию классов для эффективного управления данными, функциями и процессами приложения.

Справочная система приложения – разработать справочную систему, которая поможет пользователям быстро ориентироваться в функционале приложения и использовать его эффективно.

```
namespace App3
```

```
{
```

```
    [XamlCompilation(XamlCompilationOptions.Compile)]
```

```

public partial class MainPag : ContentPage
{
    public DateTime SelectedDate { get; set; }
    private DateTime? departureDate;
    private DateTime? returnDate;

    public MainPag()
    {
        InitializeComponent();
        NavigationPage.SetHasNavigationBar(this, false);
        // Подписка на сообщение для обновления количества пассажиров и класса
        MessagingCenter.Subscribe<Page1, string>(this,
"UpdatePassengerClassLabel", (sender, arg) =>
        {
            PassengerClassLabel.Text = arg;
        });
        //FillDatabase();
        // Заполнение базы данных, если она еще не заполнена
        if (!App.Db.GetFlights().Any())
        {
            FillDatabase(); // Заполнение базы данных рейсами
        }

        LoadPopularDestinations();
    }

    private void LoadPopularDestinations(string departureFilter = "", string
arrivalFilter = "", bool bagsIncluded = false, bool returnTrip = false, int
passengers = 1, string serviceClass = "Эконом")
    {
        var flights = App.Db.GetFlights(); // Получение данных о перелетах из
базы данных

        // Очистка существующих элементов перед добавлением новых
        PopularDestinationsContainer.Children.Clear();

        // Коэффициенты для разных классов обслуживания
        var classCoefficients = new Dictionary<string, decimal>
        {
            { "Эконом", 1.0m },
            { "Комфорт", 1.5m },
            { "Бизнес", 2.0m },
            { "Первый класс", 2.5m }
        };

        // Фильтрация рейсов
        var filteredFlights = flights.Where(f =>
            (string.IsNullOrEmpty(departureFilter) ||
f.DepartureCountry.IndexOf(departureFilter, StringComparison.OrdinalIgnoreCase) >=
0) &&
            (string.IsNullOrEmpty(arrivalFilter) ||
f.ArrivalCountry.IndexOf(arrivalFilter, StringComparison.OrdinalIgnoreCase) >= 0) &&
            (!departureDate.HasValue || f.DepartureDate.Date ==
departureDate.Value.Date) &&
            (!returnDate.HasValue || f.ArrivalDate.Date ==
returnDate.Value.Date) &&
            (!bagsIncluded || f.BagsIncluded) &&
            (!returnTrip || f.ReturnTrip) &&
            f.ServiceClass == serviceClass &&
            f.SeatCount >= passengers);

        foreach (var flight in filteredFlights)
        {
            var adjustedPrice = flight.Price * classCoefficients[serviceClass] *
passengers;

```

```

var frame = new Frame
{
    BackgroundColor = Color.FromHex("#323232"),
    CornerRadius = 10,
    Padding = 10,
    WidthRequest = 100,
    HeightRequest = 150,
    Margin = new Thickness(0, 0, 10, 0)
};

var stackLayout = new StackLayout();

// Добавление изображения аэропорта вылета
var image = new Image
{
    Source = flight.ImgDepartureAirport,
    Aspect = Aspect.AspectFit,
    VerticalOptions = LayoutOptions.CenterAndExpand
};

// Обработчик нажатия на изображение
var tapGestureRecognizer = new TapGestureRecognizer();
tapGestureRecognizer.Tapped += async (s, e) =>
{
    var modalPage = new FlightDetailsPage(flight);
    await Navigation.PushModalAsync(modalPage);
};
image.GestureRecognizers.Add(tapGestureRecognizer);

stackLayout.Children.Add(image);

// Добавление метки с названием аэропорта прилета
stackLayout.Children.Add(new Label
{
    Text = flight.ArrivalCountry,
    TextColor = Color.White,
    HorizontalOptions = LayoutOptions.Center,
    VerticalOptions = LayoutOptions.End
});

// Добавление метки с ценой или другой информацией
stackLayout.Children.Add(new Label
{
    Text = $"от {adjustedPrice} Br", // Измененная цена в
зависимости от класса и количества пассажиров
    TextColor = Color.White,
    HorizontalOptions = LayoutOptions.Center,
    VerticalOptions = LayoutOptions.End
});

frame.Content = stackLayout;
PopularDestinationsContainer.Children.Add(frame);
}

private void FillDatabase()
{
    var flights = new[]
    {
        new Flight
        {
            DepartureDate = DateTime.Now,
            ArrivalDate = DateTime.Now.AddHours(3.5),
            DepartureAirport = "Аэропорт Минск",

```

```

        ArrivalAirport = "Аэропорт Шарль де Голль",
        FlightTime = 3.5f,
        ServiceClass = "Эконом",
        ImgDepartureAirport =
"https://s16.stc.yc.kpcdn.net/share/i/12/13257320/wr-960.webp",
        ImgArrivalAirport =
"https://www.deutschland.de/sites/default/files/styles/image_carousel_mobile/public/
media/image/TdT_12032020_LeFigaro_Paris.jpg?itok=5qEkAclv",
        DepartureCountry = "Беларусь",
        BagsIncluded = true,
        ReturnTrip = false,
        ArrivalCountry = "Франция",
        Price = 3750.0m
    },
    foreach (var flight in flights)
    {
        App.Db.SaveFlights(flight);
    }

    LoadPopularDestinations(); // Обновление отображения после заполнения
базы данных
}

private void OnFilterTextChanged(object sender, TextChangedEventArgs e)
{
    LoadPopularDestinations(DepartureFilterEntry.Text,
ArrivalFilterEntry.Text);
}

private async void OnDatesTapped(object sender, EventArgs e)
{
    // Загрузка сохраненных дат, если они есть
    if (Application.Current.Properties.ContainsKey("DepartureDate"))
    {
        departureDate =
(DateTime)Application.Current.Properties["DepartureDate"];
    }
    else
    {
        departureDate = DateTime.Now;
    }

    if (Application.Current.Properties.ContainsKey("ReturnDate"))
    {
        returnDate = (DateTime)Application.Current.Properties["ReturnDate"];
    }
    else
    {
        returnDate = DateTime.Now.AddDays(1);
    }

    var departureDatePicker = new DatePicker
    {
        Date = departureDate.Value,
        MinimumDate = DateTime.Now,
        TextColor = Color.White,
        BackgroundColor = Color.FromHex("#1c1c1e"),
        Format = "dd MMMM yyyy"
    };

    var returnDatePicker = new DatePicker
    {
        Date = returnDate.Value,
        MinimumDate = departureDate.Value,
        TextColor = Color.White,

```

```

        BackgroundColor = Color.FromHex("#1c1c1e"),
        Format = "dd MMMM yyyy"
    };

    departureDatePicker.DateSelected += (s, args) =>
    {
        if (returnDatePicker.Date < departureDatePicker.Date)
        {
            returnDatePicker.Date = departureDatePicker.Date;
        }
        returnDatePicker.MinimumDate = departureDatePicker.Date;
    };

    var acceptButton = new Button
    {
        Text = "Сохранить",
        BackgroundColor = Color.FromHex("#3c8302"),
        TextColor = Color.White,
        CornerRadius = 10,
        FontSize = 18,
        TextTransform = TextTransform.None,
        FontFamily = "Roboto-Medium",
        Margin = new Thickness(0, 40, 0, 0)
    };
    acceptButton.Clicked += async (s, args) =>
    {
        // Сохранение выбранных дат
        Application.Current.Properties["DepartureDate"] =
departureDatePicker.Date;
        Application.Current.Properties["ReturnDate"] =
returnDatePicker.Date;
        await Application.Current.SavePropertiesAsync();

        // Обновление фильтрации рейсов по выбранным датам
        departureDate = departureDatePicker.Date;
        returnDate = returnDatePicker.Date;
        LoadPopularDestinations(DepartureFilterEntry.Text,
ArrivalFilterEntry.Text);

        await Navigation.PopModalAsync();
    };

    var stackLayout = new StackLayout
    {
        BackgroundColor = Color.FromHex("#1c1c1e"),
        Padding = new Thickness(20),
        VerticalOptions = LayoutOptions.CenterAndExpand,
        Children = {
            new Label { Text = "Дата отлета", TextColor = Color.White },
            departureDatePicker,
            new Label { Text = "Дата прилета", TextColor = Color.White,
Margin = new Thickness(0, 20, 0, 0) },
            returnDatePicker,
            acceptButton
        }
    };

    var modalPage = new ContentPage
    {
        Content = new Frame
        {
            Content = stackLayout,
            BackgroundColor = Color.FromHex("#1c1c1e"),
            CornerRadius = 20, // Закругленные углы формы

```

```

        Margin = new Thickness(40, 200) // Увеличенный отступ для
уменьшения размера формы
    },
    BackgroundColor = Color.FromHex("#242424"),
    Title = "Выберите даты"
};

    await Navigation.PushModalAsync(modalPage);
}

private async void OnPassengersTapped(object sender, EventArgs e)
{
    var modalPage = new Page1();
    modalPage.Disappearing += (s, args) =>
    {
        if (Application.Current.Properties.ContainsKey("Passengers"))
        {
            int passengers =
(int)Application.Current.Properties["Passengers"];
            string serviceClass =
Application.Current.Properties["SelectedClass"].ToString();
            LoadPopularDestinations(DepartureFilterEntry.Text, bagsIncluded:
false, returnTrip: false, passengers: passengers, serviceClass: serviceClass);
        }
    };
    await Navigation.PushModalAsync(modalPage);
}

private async void OnFiltersTapped(object sender, EventArgs e)
{
    var modalPage = new FiltersPage();
    modalPage.Disappearing += (s, args) =>
    {
        bool bagsIncluded =
Application.Current.Properties.ContainsKey("Bagas") &&
(bool)Application.Current.Properties["Bagas"];
        bool returnTrip =
Application.Current.Properties.ContainsKey("BothWays") &&
(bool)Application.Current.Properties["BothWays"];
        LoadPopularDestinations(DepartureFilterEntry.Text,
ArrivalFilterEntry.Text, bagsIncluded, returnTrip);
    };
    await Navigation.PushModalAsync(modalPage);
}

private async void OnProfileButtonClicked(object sender, EventArgs e)
{
    if (Application.Current.Properties.ContainsKey("username"))
    {
        // Создание новой страницы профиля
        var username = Application.Current.Properties["username"] as string;
        if (!string.IsNullOrEmpty(username))
        {
            Page2 profilePage = new Page2(username);
            // Асинхронный переход на новую страницу
            await Navigation.PushAsync(profilePage);
        }
    }
    else
    {
        // Переход на страницу профиля без имени пользователя
        Page2 profilePage = new Page2();
        await Navigation.PushAsync(profilePage);
    }
}
}

```

```

private async void OnMainButtonClicked(object sender, EventArgs e)
{
    // Создание новой страницы профиля
    MainPag MainPage = new MainPag();

    // Асинхронный переход на новую страницу
    await Navigation.PushAsync(MainPage);
}
private async void OnFavorite(object sender, EventArgs e)
{
    // Создание новой страницы профиля
    var favoritesPage = new FavoriteFlightsPage();
    await Navigation.PushAsync(favoritesPage);
}
}
}

```

3.4 Функциональное тестирование

Функциональное тестирование является важной частью процесса обеспечения качества программного обеспечения. Оно направлено на проверку функциональности системы с целью убедиться, что она соответствует требованиям и ожиданиям пользователей. Прежде чем перейти к разработке тест-кейсов, необходимо провести анализ требований, понять функциональность приложения и определить основные сценарии использования. Это позволит сосредоточиться на наиболее важных и критических аспектах приложения. Подходящие тест-кейсы должны быть разработаны для каждого сценария использования, чтобы проверить, что функции приложения работают правильно и соответствуют ожиданиям пользователей.

Таблица 3.6.1 – Тест-кейс для проверки функции авторизации пользователя

Функция	Шаги воспроизведения	Результат
Авторизация с недействительными данными	Ввод логина и пароля, нажатие на кнопку «Войти»	Ожидаемый результат: сообщение о вводе неверных данных
		Фактический результат: сообщение о вводе неверных данных
Авторизация с действительными данными	1. Заполните поля данными: – логин «11»; – пароль: «11»; 2. Нажать на кнопку «Войти»	Ожидаемый результат: переход на главную форму
		Фактический результат: переход на главную форму

На рисунке 3.1 – изображен вход в систему с некорректными данными

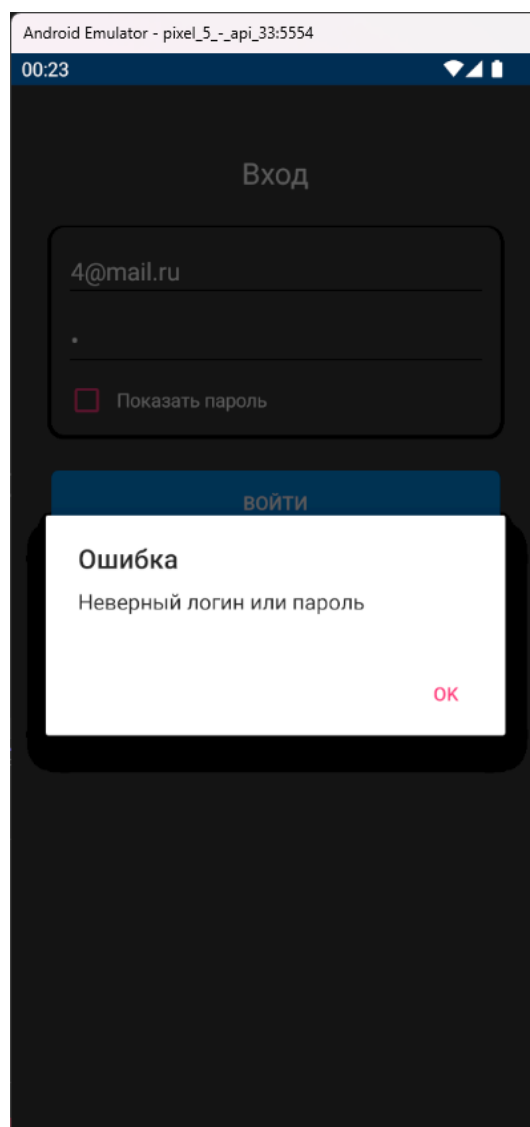


Рисунок 3.1 – Результат тест-кейса регистрации при некорректных данных

На рисунке 3.2 – изображен вход в систему с корректными данными

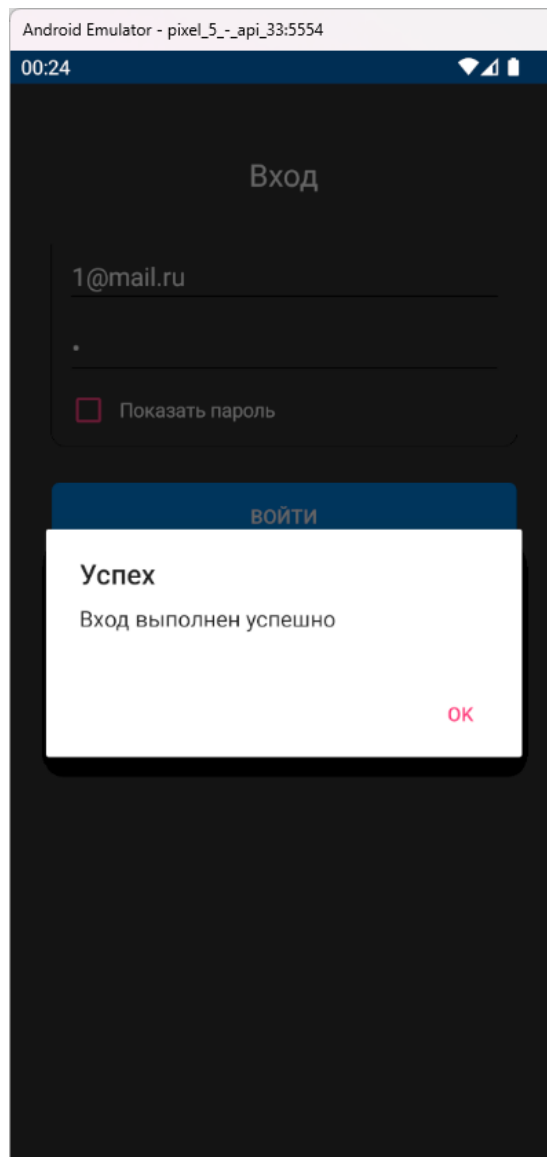


Рисунок 3.2 – Результат тест-кейса регистрации при корректных данных

4 Применение

4.1 Общие сведения

Целью разработки мобильного приложения «Калькулятор расчёта стоимости авиабилетов авиакомпании «Белавиа»» является обеспечение пользователей удобным и эффективным инструментом для поиска и расчета стоимости авиабилетов на рейсы данной авиакомпании. Приложение создано с целью упростить процесс выбора авиабилетов, предоставив пользователям быстрый доступ к актуальной информации о доступных рейсах, ценах на билеты и условиях перелета.

Для установки мобильного приложения «Калькулятор расчёта стоимости авиабилетов авиакомпании «Белавиа»» на мобильное устройство необходимо:

- перейти в магазин приложений на мобильном устройстве (например, App Store для iOS или Google Play для Android);
- в поисковой строке магазина приложений ввести название приложения «Калькулятор расчёта стоимости авиабилетов «Белавиа»»;
- нажать на кнопку «Установить» или «Скачать», в зависимости от операционной системы вашего устройства.

После установки мобильного приложения «Калькулятор расчёта стоимости авиабилетов «Белавиа»» на мобильное устройство, для его запуска необходимо:

- найти значок приложения на экране вашего мобильного устройства;
- нажать на значок приложения «Калькулятор расчёта стоимости авиабилетов «Белавиа»»;
- после этого приложение будет запущено, и вы окажетесь на главном экране приложения, готовом к использованию.

4.2 Назначение мобильного приложения

Мобильное приложение «Калькулятор расчёта стоимости авиабилетов авиакомпании «Белавиа»» разработано для обеспечения пользователей удобным и эффективным инструментом для поиска, выбора и расчета стоимости авиабилетов на рейсы данной авиакомпании. Главное назначение приложения заключается в предоставлении пользователям быстрого доступа к информации о доступных рейсах, ценах на билеты, а также возможности расчета стоимости перелета с учетом различных параметров.

Целевой аудиторией мобильного приложения являются:

- путешественники, планирующие поездки с использованием авиаперевозок;
- пользователи, ищущие выгодные предложения и акции на авиабилеты;

– люди, часто путешествующие по работе или в личных целях.

Для обеспечения безопасности данных пользователей мобильное приложение использует современные методы шифрования для защиты конфиденциальной информации, такой как персональные данные и данные о платежах.

Мобильное приложение применяется в сфере авиационных перевозок и решает следующие задачи:

- поиск доступных рейсов и билетов на авиарейсы авиакомпании «Белавиа»;

- расчет стоимости перелета с учетом различных параметров, таких как класс обслуживания, выбор места, багаж и другие дополнительные услуги.

Основным ограничением мобильного приложения является ограниченный функционал, который ограничивается только поиском и расчетом стоимости билетов авиакомпании «Белавиа». Приложение не предоставляет возможности бронирования билетов или управления бронированиями.

Диалог с пользователем осуществляется через интуитивно понятный и легко доступный пользовательский интерфейс. Пользователь может вводить информацию с помощью элементов управления, таких как текстовые поля, кнопки выбора, и другие интерактивные элементы, обеспечивая комфортное и эффективное взаимодействие с приложением.

Заключение

В ходе разработки мобильного приложения для авиакомпании «Белавиа» была поставлена задача обеспечить пользователям возможность быстрого поиска доступных рейсов и расчета стоимости авиабилетов. Для достижения этой цели было разработано программное средство, реализующее функции поиска рейсов и расчета стоимости авиабилетов на основании различных параметров.

В качестве основного инструмента разработки была выбрана платформа Xamarin с использованием языка программирования C#. В ходе выполнения задачи были реализованы все функции, описанные в техническом задании.

Разработанные функции включают:

- поиск доступных рейсов;
- расчет стоимости авиабилетов;
- сохранения авиабилетов;
- интуитивно понятный пользовательский интерфейс.

Преимуществами разработанного приложения являются удобный графический интерфейс, быстрый поиск информации и автоматический расчет стоимости билетов. Однако, ограничением может быть невозможность бронирования билетов напрямую через приложение.

Для дальнейшего улучшения приложения можно рассмотреть возможность добавления новых функций, таких как бронирование билетов, а также оптимизацию интерфейса для более удобного использования на разных устройствах.

Список использованных источников

- 1 Воздушный транспорт и авиаперевозки: Учебное пособие – Иванов И.И., Петров П.П. – М.: Издательство «Авиация», 2019.
- 2 Основы пилотирования и авиационной безопасности – Сидоров А.Н., Козлов В.П. – СПб.: Издательство «Пилот», 2018.
- 3 Авиационная медицина – Казаков К.К. – М.: Издательство «Воздух», 2017.
- 4 Техническое обслуживание и ремонт воздушных судов – Громов Г.Г., Смирнов С.С. – СПб.: Издательство «Авиатехника», 2020.
- 5 Авиационное право – Жукова Н.П. – М.: Издательство «Право и авиация», 2019.
- 6 История авиации – Михайлов И.И. – М.: Издательство «Авиация XXI века», 2018.
- 7 Авиационный маркетинг и менеджмент – Попов В.В., Лебедев Д.С. – М.: Издательство «Авиамаркет», 2017.
- 8 Авиационная психология и человеческий фактор – Соколова Е.А. – СПб.: Издательство «Авиация и психология», 2020.
- 9 LINQ [Электронный ресурс]. – Режим доступа : <https://learn.microsoft.com/ru-ru/dotnet/csharp/linq/>. – Дата доступа 28.05.2024.
- 10 C# [Электронный ресурс]. – Режим доступа : <https://learn.microsoft.com/ru-ru/dotnet/csharp/>. – Дата доступа : 26.05.2024.

Приложение А

Текст программных модулей

App.xaml:

```
<?xml version="1.0" encoding="utf-8" ?>
<Application xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="App3.App">
    <Application.Resources>

    </Application.Resources>
</Application>
```

App.xaml.cs:

```
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using System.IO;
namespace App3
{
    public partial class App : Application
    {
        private static Dtab db;
        public static Dtab Db
        {
            get
            {
                if (db == null)

                    db = new
Dtab(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicati
onData), "db.sqlite3"));
                return db;
            }
        }

        public App()
        {
            InitializeComponent();

            //MainPage = new AdminPanel2();
            MainPage = new NavigationPage(new MainPag());
        }

        protected override void OnStart()
        {
        }

        protected override void OnSleep()
        {
            // Сохраняем все данные при переходе приложения в режим ожидания
            Application.Current.SavePropertiesAsync();
            if (Application.Current.Properties.ContainsKey("username"))
            {
                var username = Application.Current.Properties["username"] as string;
                if (!string.IsNullOrEmpty(username))
                {
                    MainPage = new NavigationPage(new Page2(username));
                }
            }
        }
    }
}
```

```

        }
    }
}

protected override void OnResume()
{
}

}

}
Dtab:
using SQLite;
using System.Collections.Generic;
using System.Linq;

namespace App3
{
    public class Dtab
    {
        private readonly SQLiteConnection conn;

        public Dtab(string path)
        {
            conn = new SQLiteConnection(path);
            conn.CreateTable<Airport>();
            conn.CreateTable<User>();
            conn.CreateTable<Flight>();
            conn.CreateTable<FlightFavorite>();
        }

        // Методы для работы с аэропортами
        public List<Airport> GetAirports()
        {
            return conn.Table<Airport>().ToList();
        }
        public List<Flight> GetFlights()
        {
            return conn.Table<Flight>().ToList();
        }

        public int SaveAiroport(Airport airport)
        {
            return conn.Insert(airport);
        }
        public int SaveFlights(Flight flight)
        {
            return conn.Insert(flight);
        }

        public int DeleteItem(Airport airport)
        {
            return conn.Delete(airport);
        }
        public int DeleteFlight(Flight flight)
        {
            return conn.Delete(flight);
        }

        // Методы для работы с пользователями
        public List<User> GetUsers()
        {
            return conn.Table<User>().ToList();
        }
    }
}

```



```

public int SaveUser(User user)
{
    return conn.Insert(user);
}

public User GetUserByUsername(string username)
{
    return conn.Table<User>().FirstOrDefault(u => u.Username == username);
}

public int DeleteUser(User user)
{
    return conn.Delete(user);
}

// Методы для работы с избранными рейсами
public bool IsFavorite(int flightId, string userId)
{
    return conn.Table<FlightFavorite>().Any(f => f.FlightId == flightId &&
f.UserId == userId);
}

public int SaveFavorite(FlightFavorite favorite)
{
    return conn.Insert(favorite);
}

public List<Flight> GetFavoriteFlights(string userId)
{
    var favoriteFlightIds = conn.Table<FlightFavorite>().Where(f => f.UserId
== userId).Select(f => f.FlightId).ToList();
    return conn.Table<Flight>().Where(f =>
favoriteFlightIds.Contains(f.Id)).ToList();
}
}
}

```

FavoriteFlightsPage.xaml:

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="App3.FavoriteFlightsPage"
    BackgroundColor="#242424"
    Title="Избранные рейсы">
    <ContentPage.Content>
        <StackLayout Padding="15">
            <Label Text="Избранные рейсы" FontSize="24" TextColor="White"
HorizontalOptions="Center" FontAttributes="Bold" />
            <ListView x:Name="FavoritesListView" ItemTapped="OnFlightTapped">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <ViewCell>
                            <StackLayout Padding="5" Orientation="Horizontal">
                                <Image Source="{Binding ImgDepartureAirport}"
HeightRequest="140" WidthRequest="70" />
                                <StackLayout Orientation="Vertical" Padding="5,0">
                                    <Label Text="{Binding DepartureCountry}"
TextColor="White" FontSize="13" />
                                    <Label Text="{Binding ArrivalCountry}"
TextColor="White" FontSize="13" />
                                </StackLayout>
                            </StackLayout>
                        </ViewCell>
                    </DataTemplate>
                </ListView>
            </ContentPage.Content>
        </StackLayout>
    </ContentPage>

```

```

        </ListView.ItemTemplate>
    </ListView>
</StackLayout>
</ContentPage.Content>
</ContentPage>

```

FavoriteFlightsPage.cs:

```

using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using System.Collections.Generic;

namespace App3
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class FavoriteFlightsPage : ContentPage
    {
        public FavoriteFlightsPage()
        {
            InitializeComponent();
            LoadFavoriteFlights();
        }

        private void LoadFavoriteFlights()
        {
            var userId = "user123"; // Идентификатор пользователя, может быть
динамическим
            var favoriteFlights = App.Db.GetFavoriteFlights(userId);
            FavoritesListView.ItemsSource = favoriteFlights;
        }

        private async void OnFlightTapped(object sender, ItemTappedEventArgs e)
        {
            var selectedFlight = e.Item as Flight;
            if (selectedFlight != null)
            {
                var flightDetailsPage = new FlightDetailsPage(selectedFlight);
                await Navigation.PushModalAsync(flightDetailsPage);
            }
        }
    }
}

```

FiltersPage.xaml:

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="App3.FiltersPage"
    BackgroundColor="#2B2B2B"
    Title="Фильтры">
    <ContentPage.Content>
        <StackLayout Padding="10" VerticalOptions="FillAndExpand">
            <!-- Верхняя панель -->
            <StackLayout Orientation="Horizontal" HorizontalOptions="FillAndExpand"
VerticalOptions="Start">
                <Image Source="ic_close.png" HeightRequest="24" WidthRequest="24"
VerticalOptions="Center"/>
                <Label Text="Фильтры" FontSize="24" TextColor="White"
HorizontalOptions="CenterAndExpand" VerticalOptions="Center" FontFamily="Roboto-
Medium"/>
            </StackLayout>

            <!-- Фильтры -->
            <StackLayout Margin="0,10,0,0" VerticalOptions="StartAndExpand">
                <Label Text="Бараж" FontSize="18" TextColor="White"
Margin="0,0,0,10" FontFamily="Roboto-Medium" />
            </StackLayout>
        </ContentPage.Content>
    </ContentPage>

```

```

        <Frame BackgroundColor="#323232" CornerRadius="8" Padding="10">
            <StackLayout Orientation="Horizontal"
HorizontalOptions="FillAndExpand" VerticalOptions="Center">
                <Label Text="Только с багажом" FontSize="16"
TextColor="White" VerticalOptions="Center"/>
                <Switch x:Name="SwitchBagas"
HorizontalOptions="EndAndExpand" VerticalOptions="Center"/>
            </StackLayout>
        </Frame>
        <Label Text="В обе стороны" FontSize="18" TextColor="White"
Margin="0,10,0,10" FontFamily="Roboto-Medium" />
        <Frame BackgroundColor="#323232" CornerRadius="8" Padding="10">
            <StackLayout Orientation="Horizontal"
HorizontalOptions="FillAndExpand" VerticalOptions="Center">
                <Label Text="Только в обе стороны" FontSize="16"
TextColor="White" VerticalOptions="Center"/>
                <Switch x:Name="SwitchBothWays"
HorizontalOptions="EndAndExpand" VerticalOptions="Center"/>
            </StackLayout>
        </Frame>
    </StackLayout>

    <!-- Нижняя кнопка -->
    <Button FontSize="18" TextTransform="None" Padding="8"
x:Name="ButtonDone" Text="Готово" BackgroundColor="#3C8323" TextColor="White"
HorizontalOptions="FillAndExpand" VerticalOptions="End" CornerRadius="7"
FontFamily="Roboto-Medium" />
</StackLayout>
</ContentPage.Content>
</ContentPage>

```

FiltersPage.cs:

```

using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

```

namespace App3

```

{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class FiltersPage : ContentPage
    {
        public FiltersPage()
        {
            InitializeComponent();

            // Подписываемся на событие Appearing для восстановления состояния при
открытии модального окна
            this.Appearing += FiltersPage_Appearing;
            // Подписываемся на событие Disappearing для сохранения состояния при
закрытии модального окна
            this.Disappearing += FiltersPage_Disappearing;

            // Подписываемся на событие изменения состояния переключателя "Багаж"
            SwitchBagas.Toggled += SwitchBagas_Toggled;
            // Подписываемся на событие изменения состояния переключателя "В обе
стороны"
            SwitchBothWays.Toggled += SwitchBothWays_Toggled;
            // Подписываемся на событие нажатия на кнопку "Готово"
            ButtonDone.Clicked += ButtonDone_Clicked;
        }

        // Метод для восстановления состояния переключателей при открытии модального
окна
        private void FiltersPage_Appearing(object sender, EventArgs e)
        {

```

```

        // Устанавливаем состояние переключателя "Багаж" при открытии модального
окна
        if (Application.Current.Properties.ContainsKey("Bagas"))
        {
            SwitchBagas.IsToggled =
(bool)Application.Current.Properties["Bagas"];
        }
        // Устанавливаем состояние переключателя "В обе стороны" при открытии
модального окна
        if (Application.Current.Properties.ContainsKey("BothWays"))
        {
            SwitchBothWays.IsToggled =
(bool)Application.Current.Properties["BothWays"];
        }
    }

    // Метод для сохранения состояния переключателей при закрытии модального
окна
    private void FiltersPage_Disappearing(object sender, EventArgs e)
    {
        // Сохраняем состояние переключателя "Багаж" при закрытии модального
окна
        Application.Current.Properties["Bagas"] = SwitchBagas.IsToggled;
        // Сохраняем состояние переключателя "В обе стороны" при закрытии
модального окна
        Application.Current.Properties["BothWays"] = SwitchBothWays.IsToggled;
    }

    // Обработчик события изменения состояния переключателя "Багаж"
    private void SwitchBagas_Toggled(object sender, ToggledEventArgs e)
    {
        // Обновляем переменную Bagas при изменении состояния переключателя
        Application.Current.Properties["Bagas"] = e.Value;
    }

    // Обработчик события изменения состояния переключателя "В обе стороны"
    private void SwitchBothWays_Toggled(object sender, ToggledEventArgs e)
    {
        // Обновляем переменную BothWays при изменении состояния переключателя
        Application.Current.Properties["BothWays"] = e.Value;
    }

    // Обработчик события нажатия на кнопку "Готово"
    private async void ButtonDone_Clicked(object sender, EventArgs e)
    {
        // Закрываем модальное окно
        await Navigation.PopModalAsync();
    }
}
}

```

Flight.cs:

```

using SQLite;
using System;

namespace App3
{
    /// <summary>
    /// Представляет полёт.
    /// </summary>
    public class Flight
    {
        [PrimaryKey, AutoIncrement]

```

```

        public int Id { get; set; } // Уникальный идентификатор полёта

        public DateTime DepartureDate { get; set; } // Дата и время вылета
        public DateTime ArrivalDate { get; set; } // Дата и время прилёта

        public string ArrivalAirport { get; set; } // Код аэропорта прилёта
        public string DepartureAirport { get; set; } // Код аэропорта вылета

        public float FlightTime { get; set; } // Продолжительность полёта в часах
        public string ServiceClass { get; set; } // Класс обслуживания (например,
эконом, бизнес)

        public string ImgArrivalAirport { get; set; } // URL изображения для
аэропорта прилёта
        public string ImgDepartureAirport { get; set; } // URL изображения для
аэропорта вылета

        // Дополнительные параметры
        public string DepartureCountry { get; set; } // Страна вылета
        public string ArrivalCountry { get; set; } // Страна прилёта
        public int SeatCount { get; set; } // Количество доступных мест
        public string CountryDescription { get; set; } // Описание страны
        public string FlightNumber { get; set; } // Номер рейса

        public bool BagsIncluded { get; set; } // Включён ли багаж
        public bool ReturnTrip { get; set; } // Обратный полёт
        public decimal Price { get; set; } // Цена перелёта
    }
}

```

FlightDetailsPage.xaml:

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="App3.FlightDetailsPage"
    BackgroundColor="#242424">
    <ContentPage.Content>
        <StackLayout Padding="20">
            <Image x:Name="ImgDepartureAirport" HeightRequest="100"
Aspect="AspectFill" />
            <Image x:Name="ImgArrivalAirport" HeightRequest="100"
Aspect="AspectFill" />
            <Label x:Name="LblDepartureAirport" TextColor="White" FontSize="15"
Margin="0,10,0,0" />
            <Label x:Name="LblArrivalAirport" TextColor="White" FontSize="15"
Margin="0,10,0,0" />
            <Label x:Name="LblFlightNumber" TextColor="White" FontSize="15"
Margin="0,10,0,0" />
            <Label x:Name="LblServiceClass" TextColor="White" FontSize="15"
Margin="0,10,0,0" />
            <Label x:Name="LblDepartureDate" TextColor="White" FontSize="15"
Margin="0,10,0,0" />
            <Label x:Name="LblArrivalDate" TextColor="White" FontSize="15"
Margin="0,10,0,0" />
            <Label x:Name="LblFlightTime" TextColor="White" FontSize="15"
Margin="0,10,0,0" />
            <Label x:Name="LblPrice" TextColor="White" FontSize="15"
Margin="0,10,0,0" />
            <Label x:Name="LblSeatCount" TextColor="White" FontSize="15"
Margin="0,10,0,0" />
            <Label x:Name="LblCountryDescription" TextColor="White" FontSize="15"
Margin="0,10,0,0" />
        </StackLayout>
    </ContentPage.Content>
</ContentPage>

```

```

        <Button Text="Сохранить в избранные" TextColor="White"
        BackgroundColor="#3c8302" Clicked="OnSaveToFavoritesClicked" />
        <Button Text="Сохранить как PDF" TextColor="White"
        BackgroundColor="#3c8302" Clicked="OnSaveAsPdfClicked" />
        <Button Text="Закрыть" TextColor="White" BackgroundColor="#323232"
        Clicked="OnCloseButtonClicked" />
    </StackLayout>
</ContentPage.Content>
</ContentPage>

```

FlightDetailsPage.cs:

```

using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using System.IO;
using System.Threading.Tasks;
using Xamarin.Essentials;
using System;

namespace App3
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class FlightDetailsPage : ContentPage
    {
        private Flight currentFlight;

        public FlightDetailsPage(Flight flight)
        {
            InitializeComponent();
            currentFlight = flight;
            BindFlightDetails(flight);
        }

        private void BindFlightDetails(Flight flight)
        {
            ImgDepartureAirport.Source = flight.ImgDepartureAirport;
            ImgArrivalAirport.Source = flight.ImgArrivalAirport;
            LblDepartureAirport.Text = $"Аэропорт вылета: {flight.DepartureAirport}
({flight.DepartureCountry})";
            LblArrivalAirport.Text = $"Аэропорт прилета: {flight.ArrivalAirport}
({flight.ArrivalCountry})";
            LblFlightNumber.Text = $"Номер рейса: {flight.FlightNumber}";
            LblServiceClass.Text = $"Класс обслуживания: {flight.ServiceClass}";
            LblDepartureDate.Text = $"Дата и время вылета: {flight.DepartureDate}";
            LblArrivalDate.Text = $"Дата и время прилета: {flight.ArrivalDate}";
            LblFlightTime.Text = $"Время полета: {flight.FlightTime} часов";
            LblPrice.Text = $"Цена: {flight.Price} Br";
            LblSeatCount.Text = $"Количество мест: {flight.SeatCount}";
            LblCountryDescription.Text = $"Описание страны:
{flight.CountryDescription}";
        }

        private async void OnSaveToFavoritesClicked(object sender, EventArgs e)
        {
            var userId = "user123"; // Идентификатор пользователя, может быть
динамическим
            if (!App.Db.IsFavorite(currentFlight.Id, userId))
            {
                var favorite = new FlightFavorite
                {
                    FlightId = currentFlight.Id,
                    UserId = userId
                };
                App.Db.SaveFavorite(favorite);
                await DisplayAlert("Успех", "Рейс сохранен в избранные", "OK");
            }
        }
    }
}

```

```

    }
    else
    {
        await DisplayAlert("Информация", "Рейс уже в избранных", "OK");
    }
}

private async void OnSaveAsPdfClicked(object sender, EventArgs e)
{
    var pdfContent = GenerateFlightDetailsPdf();
    var fileName = $"{currentFlight.FlightNumber}.pdf";

    var file = Path.Combine(FileSystem.CacheDirectory, fileName);
    File.WriteAllText(file, pdfContent);

    await Share.RequestAsync(new ShareFileRequest
    {
        Title = "Сохранить как PDF",
        File = new ShareFile(file)
    });
}

private string GenerateFlightDetailsPdf()
{
    // Здесь мы создаем содержимое PDF
    return $"@
    Рейс {currentFlight.FlightNumber}
    Аэропорт вылета: {currentFlight.DepartureAirport}
    ({currentFlight.DepartureCountry})
    Аэропорт прилета: {currentFlight.ArrivalAirport}
    ({currentFlight.ArrivalCountry})
    Дата и время вылета: {currentFlight.DepartureDate}
    Дата и время прилета: {currentFlight.ArrivalDate}
    Время полета: {currentFlight.FlightTime} часов
    Класс обслуживания: {currentFlight.ServiceClass}
    Количество мест: {currentFlight.SeatCount}
    Цена: {currentFlight.Price} Br
    Описание страны: {currentFlight.CountryDescription}
    ";
}

private async void OnCloseButtonClicked(object sender, EventArgs e)
{
    await Navigation.PopModalAsync();
}
}
}

```

Login.xaml:

```

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="App3.Login"
    BackgroundColor="#2B2B2B">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*" />
            <RowDefinition Height="Auto" />
        </Grid.RowDefinitions>

        <StackLayout Padding="30" VerticalOptions="CenterAndExpand" Grid.Row="0">
            <Label Text="Вход" TextColor="White" FontSize="Large"
                HorizontalOptions="Center" Margin="0,20,0,20"/>
            <Frame BackgroundColor="#333" CornerRadius="10" Padding="10">

```

```

        <StackLayout>
            <Entry x:Name="usernameEntry" Placeholder="Логин"
TextColor="White" PlaceholderColor="Gray"/>
            <Entry x:Name="passwordEntry" Placeholder="Пароль"
IsPassword="True" TextColor="White" PlaceholderColor="Gray"/>
            <StackLayout Orientation="Horizontal">
                <CheckBox x:Name="showPasswordCheckBox"
CheckedChanged="OnShowPasswordCheckBoxChanged"/>
                <Label Text="Показать пароль" TextColor="White"
VerticalOptions="Center"/>
            </StackLayout>
        </StackLayout>
    </Frame>
    <Button Text="Войти" TextColor="White" BackgroundColor="#1E88E5"
CornerRadius="5" Margin="0,20,0,0" Clicked="OnLoginButtonClicked"/>
    <Label Text="Нет аккаунта?" TextDecorations="Underline"
TextColor="White" HorizontalOptions="Center" Margin="20,20,20,0">
        <Label.GestureRecognizers>
            <TapGestureRecognizer Tapped="OnRegisterLabelTapped"/>
        </Label.GestureRecognizers>
    </Label>
</StackLayout>
</Grid>
</ContentPage>

```

Login.cs:

```

using System;
using System.Security.Cryptography;
using System.Text;
using System.Text.RegularExpressions;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace App3
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class Login : ContentPage
    {
        public Login()
        {
            NavigationPage.SetHasNavigationBar(this, false);
            InitializeComponent();
            passwordEntry.IsPassword = true;
        }

        private async void OnLoginButtonClicked(object sender, EventArgs e)
        {
            string username = usernameEntry.Text;
            string password = passwordEntry.Text;

            if (!IsValidEmail(username))
            {
                await DisplayAlert("Ошибка", "Неверный формат email", "OK");
                return;
            }

            User user = App.Db.GetUserByUsername(username);
            if (user == null || !VerifyPassword(password, user.PasswordHash))
            {
                await DisplayAlert("Ошибка", "Неверный логин или пароль", "OK");
                return;
            }

            // Сохранение имени пользователя в свойствах приложения

```



```

        Application.Current.Properties["Username"] = username;
        await Application.Current.SavePropertiesAsync();

        await DisplayAlert("Успех", "Вход выполнен успешно", "OK");
        // Переход на страницу профиля после успешного входа
        await Navigation.PushAsync(new Page2(username));
    }

    private bool IsValidEmail(string email)
    {
        string emailPattern = @"^[^@\s]+@[^@\s]+\.[^@\s]+$";
        return Regex.IsMatch(email, emailPattern);
    }

    private bool VerifyPassword(string password, string storedHash)
    {
        using (var sha256 = SHA256.Create())
        {
            byte[] bytes = sha256.ComputeHash(Encoding.UTF8.GetBytes(password));
            StringBuilder builder = new StringBuilder();
            for (int i = 0; i < bytes.Length; i++)
            {
                builder.Append(bytes[i].ToString("x2"));
            }
            return builder.ToString() == storedHash;
        }
    }

    private void OnShowPasswordCheckBoxChanged(object sender,
        CheckedChangedEventArgs e)
    {
        passwordEntry.IsPassword = !e.Value;
    }

    private async void OnRegisterLabelTapped(object sender, EventArgs e)
    {
        await Navigation.PushAsync(new RegisterPage());
    }
}

```

MainPag.xaml:

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="App3.MainPag"
    BackgroundColor="#242424">
    <ContentPage.Content>
        <StackLayout Padding="15">
            <!-- Заголовок -->
            <Label Text="Поиск авиабилетов" FontSize="24" TextColor="White"
                HorizontalOptions="Center" FontAttributes="Bold" />

            <!-- Откуда и Куда в закругленных боксах -->
            <StackLayout Orientation="Vertical" Spacing="10">
                <Frame BackgroundColor="#323232" CornerRadius="10" Padding="10"
                    HorizontalOptions="FillAndExpand">
                    <Entry x:Name="DepartureFilterEntry" Placeholder="Откуда"
                        TextColor="White" PlaceholderColor="Gray" TextChanged="OnFilterTextChanged" />
                </Frame>
                <Frame BackgroundColor="#323232" CornerRadius="10" Padding="10"
                    HorizontalOptions="FillAndExpand">
                    <Entry x:Name="ArrivalFilterEntry" Placeholder="Куда"
                        TextColor="White" PlaceholderColor="Gray" TextChanged="OnFilterTextChanged" />
                </Frame>
            </StackLayout>
        </ContentPage.Content>
    </ContentPage>

```

```

        </Frame>
    </StackLayout>

    <!-- Даты, Количество пассажиров и Класс, Фильтры -->
    <StackLayout Orientation="Horizontal" Spacing="10" Margin="0,0,0,0">
        <Frame BackgroundColor="#323232" CornerRadius="10" Padding="10"
HorizontalOptions="FillAndExpand">
            <StackLayout Orientation="Horizontal">
                <Image Source="Calendar.png" HeightRequest="30"
WidthRequest="30" />
                <Label Text="Даты" TextColor="White"
VerticalOptions="Center" Margin="0,0,0,0" FontFamily="Roboto-Medium">
                    <Label.GestureRecognizers>
                        <TapGestureRecognizer Tapped="OnDatesTapped" />
                    </Label.GestureRecognizers>
                </Label>
            </StackLayout>
        </Frame>

        <Frame BackgroundColor="#323232" CornerRadius="10" Padding="10"
Margin="0,0,0,0" HorizontalOptions="FillAndExpand">
            <StackLayout Orientation="Horizontal">
                <Image Source="person.jpg" HeightRequest="30"
WidthRequest="30" />
                <Label x:Name="PassengerClassLabel" Text="1, эконом"
TextColor="White" VerticalOptions="Center" Margin="0,0,0,0" FontFamily="Roboto-
Medium">
                    <Label.GestureRecognizers>
                        <TapGestureRecognizer Tapped="OnPassengersTapped" />
                    </Label.GestureRecognizers>
                </Label>
            </StackLayout>
        </Frame>

        <Frame BackgroundColor="#323232" CornerRadius="10" Padding="10"
Margin="0,0,0,0" HorizontalOptions="FillAndExpand">
            <Label Text="Фильтры" TextColor="White" VerticalOptions="Center"
HorizontalOptions="Center" FontFamily="Roboto-Medium">
                <Label.GestureRecognizers>
                    <TapGestureRecognizer Tapped="OnFiltersTapped" />
                </Label.GestureRecognizers>
            </Label>
        </Frame>
    </StackLayout>

    <!-- Популярные направления -->
    <Label Text="Популярные направления" FontSize="Medium" TextColor="White"
Margin="10,20,0,10" />
    <ScrollView Orientation="Horizontal">
        <StackLayout x:Name="PopularDestinationsContainer"
Orientation="Horizontal">
            <!-- Элементы добавляются динамически из кода -->
        </StackLayout>
    </ScrollView>

    <!-- Кнопка "Показать всё" -->
    <Button Text="Показать всё" TextColor="White" BackgroundColor="#323232"
CornerRadius="10" Margin="10,20,10,0" FontFamily="Roboto-Medium"
TextTransform="None" />

    <!-- Текст "Спасибо что пользуетесь нашим приложением" -->
    <Label Text="Спасибо что пользуетесь нашим приложением"
TextColor="White" HorizontalOptions="Center" Margin="10,20,10,10" />
    <Image Source="Love.png" Aspect="AspectFit"
VerticalOptions="CenterAndExpand" HeightRequest="50" WidthRequest="50" />

```

```

        <StackLayout>
            <!-- Добавляем ваш код здесь -->
            <Grid BackgroundColor="Transparent" Padding="0"
HorizontalOptions="Fill">
                <Grid.RowDefinitions>
                    <RowDefinition Height="*" />
                    <RowDefinition Height="70" />
                </Grid.RowDefinitions>
                <!-- Размещаем кнопки в нижней строке Grid -->
                <StackLayout Grid.Row="1" Orientation="Horizontal"
BackgroundColor="#323232" Padding="0">
                    <StackLayout Margin="5,5,12,5">
                        <ImageButton Source="Plane.png"
BackgroundColor="Transparent" HeightRequest="30" WidthRequest="100"
Clicked="OnMainButtonClicked" />
                        <Label Text="Авиабилеты" TextColor="White"
HorizontalTextAlignment="Center" />
                    </StackLayout>
                    <StackLayout Margin="5,5,12,5">
                        <ImageButton Source="Calendar.png"
BackgroundColor="Transparent" HeightRequest="30" WidthRequest="100"
Clicked="OnFavorite" />
                        <Label Text="Мои билеты" TextColor="White"
HorizontalTextAlignment="Center" />
                    </StackLayout>
                    <StackLayout Margin="5,5,12,5">
                        <ImageButton Source="Person.png"
BackgroundColor="Transparent" HeightRequest="30" WidthRequest="100"
Clicked="OnProfileButtonClicked" />
                        <Label Text="Профиль" TextColor="White"
HorizontalTextAlignment="Center" />
                    </StackLayout>
                </StackLayout>
            </Grid>
        </StackLayout>
    </StackLayout>
</ContentPage.Content>
</ContentPage>

```

Main.pag.cs:

```

using System;
using System.Linq;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using System.Threading.Tasks;
using System.Collections.Generic;

```

namespace App3

```

{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class MainPag : ContentPage
    {
        public DateTime SelectedDate { get; set; }
        private DateTime? departureDate;
        private DateTime? returnDate;

        public MainPag()
        {

```

```

InitializeComponent();
NavigationPage.SetHasNavigationBar(this, false);
// Подписка на сообщение для обновления количества пассажиров и класса
MessagingCenter.Subscribe<Page1, string>(this, "UpdatePassengerClassLabel", (sender,
arg) =>
{
    PassengerClassLabel.Text = arg;
});
//FillDatabase();
// Заполнение базы данных, если она еще не заполнена
if (!App.Db.GetFlights().Any())
{
    FillDatabase(); // Заполнение базы данных рейсами
}

LoadPopularDestinations();
}

private void LoadPopularDestinations(string departureFilter = "", string arrivalFilter = "",
bool bagsIncluded = false, bool returnTrip = false, int passengers = 1, string serviceClass =
"Эконом")
{
    var flights = App.Db.GetFlights(); // Получение данных о перелетах из базы данных

    // Очищение существующих элементов перед добавлением новых
    PopularDestinationsContainer.Children.Clear();

    // Коэффициенты для разных классов обслуживания
    var classCoefficients = new Dictionary<string, decimal>
    {
        { "Эконом", 1.0m },
        { "Комфорт", 1.5m },
        { "Бизнес", 2.0m },
        { "Первый класс", 2.5m }
    };

    // Фильтрация рейсов
    var filteredFlights = flights.Where(f =>
        (string.IsNullOrEmpty(departureFilter) || f.DepartureCountry.IndexOf(departureFilter,
StringComparison.OrdinalIgnoreCase) >= 0) &&
        (string.IsNullOrEmpty(arrivalFilter) || f.ArrivalCountry.IndexOf(arrivalFilter,
StringComparison.OrdinalIgnoreCase) >= 0) &&
        (!departureDate.HasValue || f.DepartureDate.Date == departureDate.Value.Date) &&
        (!returnDate.HasValue || f.ArrivalDate.Date == returnDate.Value.Date) &&
        (!bagsIncluded || f.BagsIncluded) &&
        (!returnTrip || f.ReturnTrip) &&
        f.ServiceClass == serviceClass &&
        f.SeatCount >= passengers);

```

```

foreach (var flight in filteredFlights)
{
    var adjustedPrice = flight.Price * classCoefficients[serviceClass] * passengers;

    var frame = new Frame
    {
        BackgroundColor = Color.FromHex("#323232"),
        CornerRadius = 10,
        Padding = 10,
        WidthRequest = 100,
        HeightRequest = 150,
        Margin = new Thickness(0, 0, 10, 0)
    };

    var stackLayout = new StackLayout();

    // Добавление изображения аэропорта вылета
    var image = new Image
    {
        Source = flight.ImgDepartureAirport,
        Aspect = Aspect.AspectFit,
        VerticalOptions = LayoutOptions.CenterAndExpand
    };

    // Обработчик нажатия на изображение
    var tapGestureRecognizer = new TapGestureRecognizer();
    tapGestureRecognizer.Tapped += async (s, e) =>
    {
        var modalPage = new FlightDetailsPage(flight);
        await Navigation.PushModalAsync(modalPage);
    };
    image.GestureRecognizers.Add(tapGestureRecognizer);

    stackLayout.Children.Add(image);

    // Добавление метки с названием аэропорта прилета
    stackLayout.Children.Add(new Label
    {
        Text = flight.ArrivalCountry,
        TextColor = Color.White,
        HorizontalOptions = LayoutOptions.Center,
        VerticalOptions = LayoutOptions.End
    });

    // Добавление метки с ценой или другой информацией
    stackLayout.Children.Add(new Label
    {
        Text = $"от {adjustedPrice} Br", // Измененная цена в зависимости от класса и
        // количества пассажиров
    });
}

```

```

        TextColor = Color.White,
        HorizontalOptions = LayoutOptions.Center,
        VerticalOptions = LayoutOptions.End
    });

    frame.Content = stackLayout;
    PopularDestinationsContainer.Children.Add(frame);
}
}

private void FillDatabase()
{
    var flights = new[]
    {
        new Flight
        {
            DepartureDate = DateTime.Now,
            ArrivalDate = DateTime.Now.AddHours(3.5),
            DepartureAirport = "Аэропорт Минск",
            ArrivalAirport = "Аэропорт Шарль де Голль",
            FlightTime = 3.5f,
            ServiceClass = "Эконом",
            ImgDepartureAirport = "https://s16.stc.yc.kpcdn.net/share/i/12/13257320/wr-960.webp",
            ImgArrivalAirport =
                "https://www.deutschland.de/sites/default/files/styles/image_carousel_mobile/public/media/i-
                mage/TdT_12032020_LeFigaro_Paris.jpg?itok=5qEkAclv",
            DepartureCountry = "Беларусь",
            BagsIncluded = true,
            ReturnTrip = false,
            ArrivalCountry = "Франция",
            Price = 3750.0m
        },
        new Flight
        {
            DepartureDate = DateTime.Now,
            ArrivalDate = DateTime.Now.AddHours(11),
            DepartureAirport = "Аэропорт Франкфурт",
            ArrivalAirport = "Аэропорт Лос-Анджелес",
            FlightTime = 11.0f,
            ServiceClass = "Бизнес",
            ImgDepartureAirport = "https://ss.sport-
                express.ru/userfiles/materials/198/1983058/volga.jpg",
            ImgArrivalAirport = "https://prosto.aero/uploads/posts/2017-
                05/1493907885_ozbeoz-com-new-york-1.jpg",
            DepartureCountry = "Германия",
            BagsIncluded = true,
            ReturnTrip = false,
            ArrivalCountry = "США",
        }
    };
}

```

```

        Price = 11500.0m
    },
    new Flight
    {
        DepartureDate = DateTime.Now,
        ArrivalDate = DateTime.Now.AddHours(2.5),
        DepartureAirport = "Аэропорт Лондон Хитроу",
        ArrivalAirport = "Аэропорт Мадрид-Барахас",
        FlightTime = 2.5f,
        ServiceClass = "Эконом",
        ImgDepartureAirport = "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcSn89GGRLXWDggcO6ETonJZ0RbDfxF2zThh1Q&s",
        ImgArrivalAirport = "https://safety-
rest.ru/upload/iblock/654/654359321481747f4495029d56b25b33.jpg",
        DepartureCountry = "Великобритания",
        ArrivalCountry = "Испания",
        BagsIncluded = true,
        ReturnTrip = true,
        Price = 2780.0m
    },
    new Flight
    {
        DepartureDate = DateTime.Now.AddDays(3), // Через три дня
        ArrivalDate = DateTime.Now.AddDays(3).AddHours(6),
        DepartureAirport = "Аэропорт Токио Нарита",
        ArrivalAirport = "Аэропорт Сингапур Чанги",
        FlightTime = 6.0f,
        ServiceClass = "Эконом",
        ImgDepartureAirport = "https://cdn2.tu-
tu.ru/image/pagetree_node_data/1/288db5b1edb540e8c89cffc32f2eb55b/",
        ImgArrivalAirport = "https://cdn2.tu-
tu.ru/image/pagetree_node_data/1/bd30a23c1874571452e0e3558127a1e6/",
        DepartureCountry = "Япония",
        BagsIncluded = false,
        ReturnTrip = false,
        ArrivalCountry = "Сингапур",
        Price = 5000.0m
    },
    new Flight
    {
        DepartureDate = DateTime.Now.AddDays(4), // Через четыре дня
        ArrivalDate = DateTime.Now.AddDays(4).AddHours(1.5),
        DepartureAirport = "Аэропорт Берлин Бранденбург",
        ArrivalAirport = "Аэропорт Амстердам Схипхол",
        FlightTime = 1.5f,
        ServiceClass = "Эконом",
        ImgDepartureAirport = "https://ss.sport-
express.ru/userfiles/materials/198/1983058/volga.jpg",

```

```

        ImgArrivalAirport = "https://cdn2.tu-
tu.ru/image/pagetree_node_data/1/9ad08d19c1d0515fb87a13493232f04b/",
        DepartureCountry = "Германия",
        ArrivalCountry = "Нидерланды",
        BagsIncluded = false,
        ReturnTrip = false,
        Price = 1400.0m
    },
    new Flight
    {
        DepartureDate = DateTime.Now.AddDays(5), // Через пять дней
        ArrivalDate = DateTime.Now.AddDays(5).AddHours(8),
        DepartureAirport = "Аэропорт Сидней",
        ArrivalAirport = "Аэропорт Гонконг",
        FlightTime = 8.0f,
        ServiceClass = "Бизнес",
        ImgDepartureAirport = "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcTp3ydPTHYy8b6b6xPZR9xrC9mh8mzVGreo_c6tLR1U7l
TCJT2I7jhxJ2gqgvrYKuCexto&usqp=CAU",
        ImgArrivalAirport =
"https://tripmydream.cc/travelhub/travel/block_gallery/10/8883/gallery_first_108883.jpg",
        DepartureCountry = "Австралия",
        BagsIncluded = true,
        ReturnTrip = true,
        ArrivalCountry = "Гонконг",
        Price = 20400.0m
    },
    new Flight
    {
        DepartureDate = DateTime.Now.AddDays(6), // Через шесть дней
        ArrivalDate = DateTime.Now.AddDays(6).AddHours(4.5),
        DepartureAirport = "Аэропорт Москва Домодедово",
        ArrivalAirport = "Аэропорт Дубай",
        FlightTime = 4.5f,
        ServiceClass = "Эконом",
        ImgDepartureAirport =
"https://st2.depositphotos.com/4744673/8121/i/450/depositphotos_81219894-stock-photo-
st-basils-cathedral-on-red.jpg",
        ImgArrivalAirport =
"https://zefirtravel.by/upload/resize_cache/medialibrary/f20/810_537_2/d26bnzeuzyy6yqtqjs
h2uzix7pe9b25e.jpg",
        DepartureCountry = "Россия",
        BagsIncluded = true,
        ReturnTrip = true,
        ArrivalCountry = "ОАЭ",
        Price = 1800.0m
    },
    new Flight

```



```

{
    DepartureDate = DateTime.Now.AddDays(7), // Через семь дней
    ArrivalDate = DateTime.Now.AddDays(7).AddHours(3),
    DepartureAirport = "Аэропорт Париж Шарль де Голь",
    ArrivalAirport = "Аэропорт Рим Фьюмичино",
    FlightTime = 3.0f,
    ServiceClass = "Бизнес",
    ImgDepartureAirport =
"https://www.deutschland.de/sites/default/files/styles/image_carousel_mobile/public/media/i
mage/TdT_12032020_LeFigaro_Paris.jpg?itok=5qEkAclv",
    ImgArrivalAirport = "https://cdn2.tu-
tu.ru/image/pagetree_node_data/1/14b7c10d74a83495f37bdf74bed51875/",
    DepartureCountry = "Франция",
    ArrivalCountry = "Италия",
    SeatCount = 180,
    CountryDescription = "Италия - страна искусства и пасты.",
    FlightNumber = "AF101",
    BagsIncluded = true,
    ReturnTrip = false,
    Price = 10000.0m
},
new Flight
{
    DepartureDate = DateTime.Now.AddDays(8), // Через восемь дней
    ArrivalDate = DateTime.Now.AddDays(8).AddHours(4),
    DepartureAirport = "Аэропорт Нью-Йорк Джон Ф. Кеннеди",
    ArrivalAirport = "Аэропорт Токио Ханеда",
    FlightTime = 14.0f,
    ServiceClass = "Первый класс",
    ImgDepartureAirport = "https://prosto.aero/uploads/posts/2017-05/1493907885_ozbeoz-
com-new-york-1.jpg",
    ImgArrivalAirport = "https://cdn2.tu-
tu.ru/image/pagetree_node_data/1/288db5b1edb540e8c89cffc32f2eb55b/",
    DepartureCountry = "США",
    ArrivalCountry = "Япония",
    SeatCount = 100,
    CountryDescription = "Япония - страна смеси традиций и современности.",
    FlightNumber = "JL001",
    BagsIncluded = true,
    ReturnTrip = true,
    Price = 62500.0m
},
new Flight
{
    DepartureDate = DateTime.Now.AddDays(9), // Через девять дней
    ArrivalDate = DateTime.Now.AddDays(9).AddHours(7),
    DepartureAirport = "Аэропорт Сеул Инчхон",
    ArrivalAirport = "Аэропорт Сидней",
    FlightTime = 10.0f,

```

```

        ServiceClass = "Бизнес",
        ImgDepartureAirport = "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcQEHLWxIGid20mlwMdlUpUhZ6W3QI09GDIYg&s",
        ImgArrivalAirport = "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcTp3ydPTHYy8b6b6xPZR9xrC9mh8mzVGreo_c6tLR1U7I
TCJT2I7jhxJ2ggqvrYKuCexto&usqp=CAU",
        DepartureCountry = "Южная Корея",
        ArrivalCountry = "Австралия",
        SeatCount = 150,
        CountryDescription = "Австралия - континент с неповторимой природой.",
        FlightNumber = "KE123",
        BagsIncluded = true,
        ReturnTrip = false,
        Price = 54000.0m
    },
    new Flight
    {
        DepartureDate = DateTime.Now.AddDays(10), // Через десять дней
        ArrivalDate = DateTime.Now.AddDays(10).AddHours(3.5),
        DepartureAirport = "Аэропорт Хельсинки",
        ArrivalAirport = "Аэропорт Осло Гардермоен",
        FlightTime = 1.5f,
        ServiceClass = "Эконом",
        ImgDepartureAirport = "https://cdn2.tu-
tu.ru/image/pagetree_node_data/1/89714e2a8a24a8bac0fa06945bc874e7/",
        ImgArrivalAirport = "https://media.vand.ru/countries/about/nor/ib2.webp",
        DepartureCountry = "Финляндия",
        ArrivalCountry = "Норвегия",
        SeatCount = 200,
        CountryDescription = "Норвегия - страна фьордов и северного сияния.",
        FlightNumber = "AY456",
        BagsIncluded = true,
        ReturnTrip = false,
        Price = 8750.0m
    },
    new Flight
    {
        DepartureDate = DateTime.Now.AddDays(11), // Через одиннадцать дней
        ArrivalDate = DateTime.Now.AddDays(11).AddHours(8),
        DepartureAirport = "Аэропорт Москва Шереметьево",
        ArrivalAirport = "Аэропорт Пекин",
        FlightTime = 8.0f,
        ServiceClass = "Бизнес",
        ImgDepartureAirport =
"https://zefirtravel.by/upload/resize_cache/medialibrary/f20/810_537_2/d26bnzeuzyy6yqtqjs
h2uzix7pe9b25e.jpg",
        ImgArrivalAirport = "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcTt9R6GМаepCy0Gi7U7vsjjOyrq3BnXukttfg&s",
        DepartureCountry = "Россия",

```

```

ArrivalCountry = "Китай",
SeatCount = 180,
CountryDescription = "Китай - страна древней культуры и современных технологий.",
FlightNumber = "SU888",
BagsIncluded = true,
ReturnTrip = true,
Price = 30000.0m
},
new Flight
{
    DepartureDate = DateTime.Now.AddDays(12), // Через двенадцать дней
    ArrivalDate = DateTime.Now.AddDays(12).AddHours(6),
    DepartureAirport = "Аэропорт Лос-Анджелес",
    ArrivalAirport = "Аэропорт Сан-Франциско",
    FlightTime = 1.5f,
    ServiceClass = "Эконом",
    ImgDepartureAirport = "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcQ0yvRLQxsj-VrrQ4laO62UaKkiZMZ_INnUpQ&s",
    ImgArrivalAirport = "https://avatars.mds.yandex.net/get-
altay/1938323/2a0000016f3bc77bc216a28fc4a405fc2181/L_height",
    DepartureCountry = "США",
    ArrivalCountry = "США",
    SeatCount = 220,
    CountryDescription = "США - страна разнообразия и возможностей.",
    FlightNumber = "UA456",
    BagsIncluded = true,
    ReturnTrip = false,
    Price = 2400.0m
},
new Flight
{
    DepartureDate = DateTime.Now.AddDays(13), // Через тринадцать дней
    ArrivalDate = DateTime.Now.AddDays(13).AddHours(9),
    DepartureAirport = "Аэропорт Дели",
    ArrivalAirport = "Аэропорт Сеул Инчхон",
    FlightTime = 7.0f,
    ServiceClass = "Первый класс",
    ImgDepartureAirport =
"https://interaffairs.ru/i/2019/01/6f2822e99990945858be376da3f56b07.jpg",
    ImgArrivalAirport = "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcQEHLWxlGid20mlwMdlUpUhz6W3Ql09GDIYg&s",
    DepartureCountry = "Индия",
    ArrivalCountry = "Южная Корея",
    SeatCount = 120,
    CountryDescription = "Южная Корея - страна технологий и трендов.",
    FlightNumber = "AI789",
    BagsIncluded = true,
    ReturnTrip = false,
    Price = 42000.0m
}

```

```

},
new Flight
{
    DepartureDate = DateTime.Now.AddDays(14), // Через четырнадцать дней
    ArrivalDate = DateTime.Now.AddDays(14).AddHours(5),
    DepartureAirport = "Аэропорт Торонто Пирсон",
    ArrivalAirport = "Аэропорт Ванкувер",
    FlightTime = 4.0f,
    ServiceClass = "Бизнес",
    ImgDepartureAirport = "https://img.tourister.ru/files/1/7/5/0/8/0/5/6/original.jpg",
    ImgArrivalAirport =
"https://yestravel.ru/upload/shop_6/8/2/5/item_825541/item_825541.jpg",
    DepartureCountry = "Канада",
    ArrivalCountry = "Канада",
    SeatCount = 150,
    CountryDescription = "Канада - страна просторов и природы.",
    FlightNumber = "AC123",
    BagsIncluded = true,
    ReturnTrip = true,
    Price = 12500.0m
},
new Flight
{
    DepartureDate = DateTime.Now.AddDays(15), // Через пятнадцать дней
    ArrivalDate = DateTime.Now.AddDays(15).AddHours(2),
    DepartureAirport = "Аэропорт Амстердам Схипхол",
    ArrivalAirport = "Аэропорт Милан Линате",
    FlightTime = 1.5f,
    ServiceClass = "Эконом",
    ImgDepartureAirport = "https://cdn2.tu-
tu.ru/image/pagetree_node_data/1/9ad08d19c1d0515fb87a13493232f04b/",
    ImgArrivalAirport = "https://cdn2.tu-
tu.ru/image/pagetree_node_data/1/14b7c10d74a83495f37bdf74bed51875/",
    DepartureCountry = "Нидерланды",
    ArrivalCountry = "Италия",
    SeatCount = 180,
    CountryDescription = "Италия - страна моды и страсти.",
    FlightNumber = "KL789",
    BagsIncluded = true,
    ReturnTrip = false,
    Price = 5750.0m
},
new Flight
{
    DepartureDate = DateTime.Now.AddDays(16), // Через шестнадцать дней
    ArrivalDate = DateTime.Now.AddDays(16).AddHours(3),
    DepartureAirport = "Аэропорт Шанхай Пудонг",
    ArrivalAirport = "Аэропорт Гонконг",
    FlightTime = 2.0f,

```

```

        ServiceClass = "Эконом",
        ImgDepartureAirport = "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcTt9R6GMaepCy0Gi7U7vsjjOyrq3BnXukttfg&s",
        ImgArrivalAirport =
"https://tripmydream.cc/travelhub/travel/block_gallery/10/8883/gallery_first_108883.jpg",
        DepartureCountry = "Китай",
        ArrivalCountry = "Гонконг",
        SeatCount = 200,
        CountryDescription = "Гонконг - мегаполис с восточным колоритом.",
        FlightNumber = "MU456",
        BagsIncluded = true,
        ReturnTrip = false,
        Price = 6000.0m
    },
    new Flight
    {
        DepartureDate = DateTime.Now.AddDays(17), // Через семнадцать дней
        ArrivalDate = DateTime.Now.AddDays(17).AddHours(4),
        DepartureAirport = "Аэропорт Лиссабон Портела",
        ArrivalAirport = "Аэропорт Барселона Эль Прат",
        FlightTime = 2.5f,
        ServiceClass = "Эконом",
        ImgDepartureAirport = "https://cdn2.tu-
tu.ru/image/pagetree_node_data/1/dd54d55f4e0c3f3c6bd43f7c54ab3731/",
        ImgArrivalAirport = "https://safety-
rest.ru/upload/iblock/654/654359321481747f4495029d56b25b33.jpg",
        DepartureCountry = "Португалия",
        ArrivalCountry = "Испания",
        SeatCount = 160,
        CountryDescription = "Испания - страна солнца и пляжей.",
        FlightNumber = "TP789",
        BagsIncluded = false,
        ReturnTrip = true,
        Price = 8400.0m
    },
    new Flight
    {
        DepartureDate = DateTime.Now.AddDays(1),
        ArrivalDate = DateTime.Now.AddDays(1).AddHours(6),
        DepartureAirport = "Аэропорт Мадрид Барахас",
        ArrivalAirport = "Аэропорт Москва Домодедово",
        FlightTime = 5.5f,
        ServiceClass = "Эконом",
        ImgDepartureAirport = "https://safety-
rest.ru/upload/iblock/654/654359321481747f4495029d56b25b33.jpg",
        ImgArrivalAirport =
"https://st2.depositphotos.com/4744673/8121/i/450/depositphotos_81219894-stock-photo-
st-basils-cathedral-on-red.jpg",
        DepartureCountry = "Испания",

```

```

        ArrivalCountry = "Россия",
        BagsIncluded = true,
        ReturnTrip = false,
        Price = 4500.0m // Converted from EUR to BYN at a rate of 2.3
    },
    new Flight
    {
        DepartureDate = DateTime.Now.AddDays(2),
        ArrivalDate = DateTime.Now.AddDays(2).AddHours(8),
        DepartureAirport = "Аэропорт Пекин",
        ArrivalAirport = "Аэропорт Торонто Пирсон",
        FlightTime = 13.0f,
        ServiceClass = "Бизнес",
        ImgDepartureAirport = "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcTt9R6GМаерCy0Gi7U7vsjjOyrq3BnXukttfg&s",
        ImgArrivalAirport = "https://img.tourister.ru/files/1/7/5/0/8/0/5/6/original.jpg",
        DepartureCountry = "Китай",
        ArrivalCountry = "Канада",
        BagsIncluded = true,
        ReturnTrip = true,
        Price = 34000.0m // Converted from CAD to BYN at a rate of 1.8
    },
    new Flight
    {
        DepartureDate = DateTime.Now.AddDays(3),
        ArrivalDate = DateTime.Now.AddDays(3).AddHours(5),
        DepartureAirport = "Аэропорт Лондон Хитроу",
        ArrivalAirport = "Аэропорт Сидней",
        FlightTime = 20.0f,
        ServiceClass = "Первый класс",
        ImgDepartureAirport = "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcSn89GGRLXWDggcO6ETonJZ0RbDfxF2zThh1Q&s",
        ImgArrivalAirport = "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcTp3ydPTHYy8b6b6xPZR9xrC9mh8mzVGreo_c6tLR1U7l
TCJT2I7jhxJ2gqgvYKuCexto&usqr=CAU",
        DepartureCountry = "Великобритания",
        ArrivalCountry = "Австралия",
        BagsIncluded = true,
        ReturnTrip = true,
        Price = 58000.0m // Converted from GBP to BYN at a rate of 2.5
    },
    new Flight
    {
        DepartureDate = DateTime.Now.AddDays(4),
        ArrivalDate = DateTime.Now.AddDays(4).AddHours(3),
        DepartureAirport = "Аэропорт Шанхай Пудонг",
        ArrivalAirport = "Аэропорт Дублин",
        FlightTime = 11.5f,
        ServiceClass = "Эконом",

```

```

        ImgDepartureAirport = "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcTt9R6GMAepCy0Gi7U7vsjjOyrq3BnXukttfg&s",
        ImgArrivalAirport = "https://www.historvius.com/images/original/531/dublin.jpg",
        DepartureCountry = "Китай",
        ArrivalCountry = "Ирландия",
        BagsIncluded = true,
        ReturnTrip = false,
        Price = 6100.0m // Converted from EUR to BYN at a rate of 2.3
    },
    new Flight
    {
        DepartureDate = DateTime.Now.AddDays(5),
        ArrivalDate = DateTime.Now.AddDays(5).AddHours(7),
        DepartureAirport = "Аэропорт Сингапур Чанги",
        ArrivalAirport = "Аэропорт Сеул Инчхон",
        FlightTime = 6.0f,
        ServiceClass = "Бизнес",
        ImgDepartureAirport = "https://cdn2.tu-
tu.ru/image/pagetree_node_data/1/bd30a23c1874571452e0e3558127a1e6/",
        ImgArrivalAirport = "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcQEHDLWxIGid20mlwMdlUpUhZ6W3QI09GDIYg&s",
        DepartureCountry = "Сингапур",
        ArrivalCountry = "Южная Корея",
        BagsIncluded = true,
        ReturnTrip = false,
        Price = 17300.0m // Converted from SGD to BYN at a rate of 1.6
    },
    new Flight
    {
        DepartureDate = DateTime.Now.AddDays(6),
        ArrivalDate = DateTime.Now.AddDays(6).AddHours(4),
        DepartureAirport = "Аэропорт Нью-Йорк Джон Ф. Кеннеди",
        ArrivalAirport = "Аэропорт Лондон Хитроу",
        FlightTime = 7.0f,
        ServiceClass = "Эконом",
        ImgDepartureAirport = "https://prosto.aero/uploads/posts/2017-
05/1493907885_ozbeoz-com-new-york-1.jpg",
        ImgArrivalAirport = "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcSn89GGRLXWDggcO6ETonJZ0RbDfxF2zThh1Q&s",
        DepartureCountry = "США",
        ArrivalCountry = "Великобритания",
        BagsIncluded = true,
        ReturnTrip = false,
        Price = 7900.0m // Converted from USD to BYN at a rate of 2.1
    },
    new Flight
    {
        DepartureDate = DateTime.Now.AddDays(7),
        ArrivalDate = DateTime.Now.AddDays(7).AddHours(3),

```

```

        DepartureAirport = "Аэропорт Токио Ханеда",
        ArrivalAirport = "Аэропорт Шанхай Пудонг",
        FlightTime = 2.5f,
        ServiceClass = "Бизнес",
        ImgDepartureAirport = "https://cdn2.tu-
tu.ru/image/pagetree_node_data/1/288db5b1edb540e8c89cffc32f2eb55b/",
        ImgArrivalAirport = "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcTt9R6GМаерCy0Gi7U7vsjjOyrq3BnXukttfg&s",
        DepartureCountry = "Япония",
        ArrivalCountry = "Китай",
        BagsIncluded = true,
        ReturnTrip = true,
        Price = 31000.0m // Converted from CNY to BYN at a rate of 4.2
    }
};

foreach (var flight in flights)
{
    App.Db.SaveFlights(flight);
}

LoadPopularDestinations(); // Обновление отображения после заполнения базы
данных
}

private void OnFilterTextChanged(object sender, TextChangedEventArgs e)
{
    LoadPopularDestinations(DepartureFilterEntry.Text, ArrivalFilterEntry.Text);
}

private async void OnDatesTapped(object sender, EventArgs e)
{
    // Загрузка сохраненных дат, если они есть
    if (Application.Current.Properties.ContainsKey("DepartureDate"))
    {
        departureDate = (DateTime)Application.Current.Properties["DepartureDate"];
    }
    else
    {
        departureDate = DateTime.Now;
    }

    if (Application.Current.Properties.ContainsKey("ReturnDate"))
    {
        returnDate = (DateTime)Application.Current.Properties["ReturnDate"];
    }
    else
    {
        returnDate = DateTime.Now.AddDays(1);
    }
}

```



```

}

var departureDatePicker = new DatePicker
{
    Date = departureDate.Value,
    MinimumDate = DateTime.Now,
    TextColor = Color.White,
    BackgroundColor = Color.FromHex("#1c1c1e"),
    Format = "dd MMMM yyyy"
};

var returnDatePicker = new DatePicker
{
    Date = returnDate.Value,
    MinimumDate = departureDate.Value,
    TextColor = Color.White,
    BackgroundColor = Color.FromHex("#1c1c1e"),
    Format = "dd MMMM yyyy"
};

departureDatePicker.DateSelected += (s, args) =>
{
    if (returnDatePicker.Date < departureDatePicker.Date)
    {
        returnDatePicker.Date = departureDatePicker.Date;
    }
    returnDatePicker.MinimumDate = departureDatePicker.Date;
};

var acceptButton = new Button
{
    Text = "Сохранить",
    BackgroundColor = Color.FromHex("#3c8302"),
    TextColor = Color.White,
    CornerRadius = 10,
    FontSize = 18,
    TextTransform = TextTransform.None,
    FontFamily = "Roboto-Medium",
    Margin = new Thickness(0, 40, 0, 0)
};

acceptButton.Clicked += async (s, args) =>
{
    // Сохранение выбранных дат
    Application.Current.Properties["DepartureDate"] = departureDatePicker.Date;
    Application.Current.Properties["ReturnDate"] = returnDatePicker.Date;
    await Application.Current.SavePropertiesAsync();

    // Обновление фильтрации рейсов по выбранным датам
    departureDate = departureDatePicker.Date;

```

```

        returnDate = returnDatePicker.Date;
        LoadPopularDestinations(DepartureFilterEntry.Text, ArrivalFilterEntry.Text);

        await Navigation.PopModalAsync();
    };

    var stackLayout = new StackLayout
    {
        BackgroundColor = Color.FromHex("#1c1c1e"),
        Padding = new Thickness(20),
        VerticalOptions = LayoutOptions.CenterAndExpand,
        Children = {
            new Label { Text = "Дата отлета", TextColor = Color.White },
            departureDatePicker,
            new Label { Text = "Дата прилета", TextColor = Color.White, Margin = new
Thickness(0, 20, 0, 0) },
            returnDatePicker,
            acceptButton
        }
    };

    var modalPage = new ContentPage
    {
        Content = new Frame
        {
            Content = stackLayout,
            BackgroundColor = Color.FromHex("#1c1c1e"),
            CornerRadius = 20, // Закругленные углы формы
            Margin = new Thickness(40, 200) // Увеличенный отступ для уменьшения
размера формы
        },
        BackgroundColor = Color.FromHex("#242424"),
        Title = "Выберите даты"
    };

    await Navigation.PushModalAsync(modalPage);
}

private async void OnPassengersTapped(object sender, EventArgs e)
{
    var modalPage = new Page1();
    modalPage.Disappearing += (s, args) =>
    {
        if (Application.Current.Properties.ContainsKey("Passengers"))
        {
            int passengers = (int)Application.Current.Properties["Passengers"];
            string serviceClass = Application.Current.Properties["SelectedClass"].ToString();
            LoadPopularDestinations(DepartureFilterEntry.Text, bagsIncluded: false, returnTrip:
false, passengers: passengers, serviceClass: serviceClass);

```

```

    }
};
await Navigation.PushModalAsync(modalPage);
}

private async void OnFiltersTapped(object sender, EventArgs e)
{
    var modalPage = new FiltersPage();
    modalPage.Disappearing += (s, args) =>
    {
        bool bagsIncluded = Application.Current.Properties.ContainsKey("Bagas") &&
(bool)Application.Current.Properties["Bagas"];
        bool returnTrip = Application.Current.Properties.ContainsKey("BothWays") &&
(bool)Application.Current.Properties["BothWays"];
        LoadPopularDestinations(DepartureFilterEntry.Text, ArrivalFilterEntry.Text,
bagsIncluded, returnTrip);
    };
    await Navigation.PushModalAsync(modalPage);
}

private async void OnProfileButtonClicked(object sender, EventArgs e)
{
    if (Application.Current.Properties.ContainsKey("username"))
    {
        // Создание новой страницы профиля
        var username = Application.Current.Properties["username"] as string;
        if (!string.IsNullOrEmpty(username))
        {
            Page2 profilePage = new Page2(username);
            // Асинхронный переход на новую страницу
            await Navigation.PushAsync(profilePage);
        }
    }
    else
    {
        // Переход на страницу профиля без имени пользователя
        Page2 profilePage = new Page2();
        await Navigation.PushAsync(profilePage);
    }
}

private async void OnMainButtonClicked(object sender, EventArgs e)
{
    // Создание новой страницы профиля
    MainPag MainPage = new MainPag();

    // Асинхронный переход на новую страницу
    await Navigation.PushAsync(MainPage);
}

```

```

private async void OnFavorite(object sender, EventArgs e)
{
    // Создание новой страницы профиля
    var favoritesPage = new FavoriteFlightsPage();
    await Navigation.PushAsync(favoritesPage);
}
}
}
Main.Page.cs:
using System;
using Xamarin.Forms;

namespace App3
{
    public partial class MainPage : ContentPage
    {
        private bool modalOpened = false; // Флаг, отслеживающий, было ли уже
открыто модальное окно
        public MainPage()
        {
            InitializeComponent();
            NavigationPage.SetHasNavigationBar(this, false);
        }

        private async void OnSkipClicked(object sender, EventArgs e)
        {
            await Navigation.PopModalAsync(); // Кнопка пропустить закрытия
модального окна
        }

        private async void OnLoginClicked(object sender, EventArgs e)
        {
            Application.Current.MainPage = new Page2();
        }

        private async void OnLaterClicked(object sender, EventArgs e)
        {
            // Переход на страницу MainPag по нажатию на кнопку Потом
            // Закрыть текущее модальное окно
            await Navigation.PopModalAsync();
        }

        protected override void OnAppearing()
        {
            base.OnAppearing();
            NavigationPage.SetHasNavigationBar(this, false);
            base.OnAppearing();

            // Открыть модальное окно только если оно еще не было открыто
            if (!modalOpened)
            {
                modalOpened = true;
            }
        }
    }
}

```

Page1.xaml:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="App3.Page1"
    BackgroundColor="#2B2B2B">
    <ContentPage.Content>
        <StackLayout Padding="20">
            <!-- Заголовок -->
            <Label Text="Пассажиры и класс" FontSize="24" TextColor="White"
HorizontalOptions="Center" Margin="0,10,0,0" />
            <BoxView HeightRequest="1" BackgroundColor="#5E5E5E" Margin="0,50,0,5"
/>

            <!-- Взрослые -->
            <Grid Padding="0, 5">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="55" />
                    <ColumnDefinition Width="30" />
                    <ColumnDefinition Width="55" />
                </Grid.ColumnDefinitions>
                <Label Text="Взрослые" FontSize="18" TextColor="White"
Grid.Column="0" VerticalOptions="Center" Margin="0,5,0,20" />
                <Label Text="12 лет и старше" FontSize="Small" TextColor="Gray"
Grid.Column="0" VerticalOptions="End" />
                <Button Text="-" BackgroundColor="#4D4D4D" TextColor="White"
Grid.Column="1" Clicked="OnDecreaseAdults" BorderRadius="30" FontSize="20" />
                <Label x:Name="AdultsLabel" Text="1" FontSize="Large"
TextColor="White" Grid.Column="2" VerticalOptions="CenterAndExpand"
HorizontalOptions="Center" />
                <Button Text="+" BackgroundColor="#2196F3" TextColor="White"
Grid.Column="3" Clicked="OnIncreaseAdults" BorderRadius="30" FontSize="20" />
            </Grid>

            <BoxView HeightRequest="1" BackgroundColor="#5E5E5E" Margin="0,5,0,5" />

            <!-- Дети -->
            <Grid Padding="0, 5">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="55" />
                    <ColumnDefinition Width="30" />
                    <ColumnDefinition Width="55" />
                </Grid.ColumnDefinitions>
                <Label Text="Дети" FontSize="18" TextColor="White" Grid.Column="0"
VerticalOptions="Center" Margin="0,5,0,20" />
                <Label Text="От 2 до 11 лет" FontSize="Small" TextColor="Gray"
Grid.Column="0" VerticalOptions="End" />
                <Button Text="-" BackgroundColor="#4D4D4D" TextColor="White"
Grid.Column="1" Clicked="OnDecreaseChildren" BorderRadius="30" FontSize="25" />
                <Label x:Name="ChildrenLabel" Text="0" FontSize="Large"
TextColor="White" Grid.Column="2" VerticalOptions="CenterAndExpand"
HorizontalOptions="Center" />
                <Button Text="+" BackgroundColor="#2196F3" TextColor="White"
Grid.Column="3" Clicked="OnIncreaseChildren" BorderRadius="30" FontSize="20" />
            </Grid>

            <BoxView HeightRequest="1" BackgroundColor="#5E5E5E" Margin="0,10,0,10"
/>

            <!-- Младенцы -->
            <Grid Padding="0, 5">
```

```

        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="55" />
            <ColumnDefinition Width="30" />
            <ColumnDefinition Width="55" />
        </Grid.ColumnDefinitions>
        <Label Text="Младенцы" FontSize="18" TextColor="White"
Grid.Column="0" VerticalOptions="Center" Margin="0,5,0,20" />
        <Label Text="Младше 2 лет, без места" FontSize="Small"
TextColor="Gray" Grid.Column="0" VerticalOptions="End" />
        <Button Text="-" BackgroundColor="#4D4D4D" TextColor="White"
Grid.Column="1" Clicked="OnDecreaseInfants" BorderRadius="30" FontSize="20" />
        <Label x:Name="InfantsLabel" Text="0" FontSize="Large"
TextColor="White" Grid.Column="2" VerticalOptions="CenterAndExpand"
HorizontalOptions="Center" />
        <Button Text="+" BackgroundColor="#2196F3" TextColor="White"
Grid.Column="3" Clicked="OnIncreaseInfants" BorderRadius="30" FontSize="20" />
    </Grid>

    <!-- Выбор класса -->
    <StackLayout Padding="0" Margin="0,40,0,0">
        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="40" />
            </Grid.ColumnDefinitions>
            <Label Text="Эконом" FontSize="Large" TextColor="White"
VerticalOptions="Center" Grid.Column="0" />
            <RadioButton x:Name="EconomyClassLabel" Grid.Column="1"
FontSize="Large" TextColor="White" CheckedChanged="OnEconomyClassSelected"
GroupName="class" />
        </Grid>
        <BoxView HeightRequest="1" BackgroundColor="#5E5E5E"
Margin="0,10,0,10" />
        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="40" />
            </Grid.ColumnDefinitions>
            <Label Text="Комфорт" FontSize="Large" TextColor="White"
VerticalOptions="Center" Grid.Column="0" />
            <RadioButton x:Name="ComfortClassLabel" Grid.Column="1"
FontSize="Large" TextColor="White" CheckedChanged="OnComfortClassSelected"
GroupName="class" />
        </Grid>
        <BoxView HeightRequest="1" BackgroundColor="#5E5E5E"
Margin="0,10,0,10" />
        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="40" />
            </Grid.ColumnDefinitions>
            <Label Text="Бизнес" FontSize="Large" TextColor="White"
VerticalOptions="Center" Grid.Column="0" />
            <RadioButton x:Name="BusinessClassLabel" Grid.Column="1"
FontSize="Large" TextColor="White" CheckedChanged="OnBusinessClassSelected"
GroupName="class" />
        </Grid>
        <BoxView HeightRequest="1" BackgroundColor="#5E5E5E"
Margin="0,10,0,10" />
        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="40" />
            </Grid.ColumnDefinitions>

```

```

        <Label Text="Первый класс" FontSize="Large" TextColor="White"
VerticalOptions="Center" Grid.Column="0" />
        <RadioButton x:Name="FirstClassLabel" Grid.Column="1"
FontSize="Large" TextColor="White" CheckedChanged="OnFirstClassSelected"
GroupName="class" />
    </Grid>
</StackLayout>

<!-- Кнопка подтверждения -->
<Button Text="Выбрать" BackgroundColor="#3C8323" TextColor="White"
Clicked="OnConfirmSelection" BorderRadius="8" Margin="0,60,0,10" />

<!-- Обновляемая надпись -->
<Label x:Name="SummaryLabel" Text="1, Эконом" FontSize="Medium"
TextColor="White" HorizontalOptions="Center" Margin="0,10,0,0" IsVisible="false"/>
</StackLayout>
</ContentPage.Content>
</ContentPage>

```

Page1.cs:

```
using Xamarin.Forms;
```

```
namespace App3
```

```
{
```

```
    public partial class Page1 : ContentPage
    {
```

```
        int adults = 1;
        int children = 0;
        int infants = 0;
        string selectedClass = "Эконом";

```

```
        public Page1()
        {
```

```
            InitializeComponent();

```

```
            // Подписываемся на события Appearing и Disappearing для сохранения и
восстановления состояния

```

```
            this.Appearing += Page1_Appearing;
            this.Disappearing += Page1_Disappearing;

```

```
            // Инициализируем надпись при запуске
            UpdateSummaryLabel();

```

```
        }

```

```
        private void Page1_Appearing(object sender, System.EventArgs e)
        {
```

```
            // Восстанавливаем состояние при открытии страницы
            if (Application.Current.Properties.ContainsKey("Adults"))
            {
```

```
                AdultsLabel.Text =
Application.Current.Properties["Adults"].ToString();
            }

```

```
            if (Application.Current.Properties.ContainsKey("Children"))
            {
```

```
                ChildrenLabel.Text =
Application.Current.Properties["Children"].ToString();
            }

```

```
            if (Application.Current.Properties.ContainsKey("Infants"))
            {
```

```
                InfantsLabel.Text =
Application.Current.Properties["Infants"].ToString();
            }

```

```
            if (Application.Current.Properties.ContainsKey("SelectedClass"))

```

```

        {
            selectedClass =
Application.Current.Properties["SelectedClass"].ToString();
            UpdateSelectedClass();
        }

        UpdateSummaryLabel();
    }

private void Page1_Disappearing(object sender, System.EventArgs e)
{
    // Сохраняем состояние при закрытии страницы
    Application.Current.Properties["Adults"] = AdultsLabel.Text;
    Application.Current.Properties["Children"] = ChildrenLabel.Text;
    Application.Current.Properties["Infants"] = InfantsLabel.Text;
    Application.Current.Properties["SelectedClass"] = selectedClass;

    // Сохраняем общее количество пассажиров
    int totalPassengers = int.Parse(AdultsLabel.Text) +
int.Parse(ChildrenLabel.Text) + int.Parse(InfantsLabel.Text);
    Application.Current.Properties["Passengers"] = totalPassengers;
}

void OnIncreaseAdults(object sender, System.EventArgs e)
{
    int adults = int.Parse(AdultsLabel.Text);
    if (adults < 9)
    {
        adults++;
        AdultsLabel.Text = adults.ToString();
        UpdateSummaryLabel();
    }
}

void OnDecreaseAdults(object sender, System.EventArgs e)
{
    int adults = int.Parse(AdultsLabel.Text);
    if (adults > 1)
    {
        adults--;
        AdultsLabel.Text = adults.ToString();
        UpdateSummaryLabel();
    }
}

void OnIncreaseChildren(object sender, System.EventArgs e)
{
    int children = int.Parse(ChildrenLabel.Text);
    if (children < 9)
    {
        children++;
        ChildrenLabel.Text = children.ToString();
        UpdateSummaryLabel();
    }
}

void OnDecreaseChildren(object sender, System.EventArgs e)
{
    int children = int.Parse(ChildrenLabel.Text);
    if (children > 0)
    {
        children--;
        ChildrenLabel.Text = children.ToString();
        UpdateSummaryLabel();
    }
}

```



```

}

void OnIncreaseInfants(object sender, System.EventArgs e)
{
    int infants = int.Parse(InfantsLabel.Text);
    if (infants < 9)
    {
        infants++;
        InfantsLabel.Text = infants.ToString();
        UpdateSummaryLabel();
    }
}

void OnDecreaseInfants(object sender, System.EventArgs e)
{
    int infants = int.Parse(InfantsLabel.Text);
    if (infants > 0)
    {
        infants--;
        InfantsLabel.Text = infants.ToString();
        UpdateSummaryLabel();
    }
}

void OnEconomyClassSelected(object sender, CheckedChangedEventArgs e)
{
    if (e.Value)
    {
        selectedClass = "Эконом";
        SaveSelectedClass();
        UpdateSummaryLabel();
    }
}

void OnComfortClassSelected(object sender, CheckedChangedEventArgs e)
{
    if (e.Value)
    {
        selectedClass = "Комфорт";
        SaveSelectedClass();
        UpdateSummaryLabel();
    }
}

void OnBusinessClassSelected(object sender, CheckedChangedEventArgs e)
{
    if (e.Value)
    {
        selectedClass = "Бизнес";
        SaveSelectedClass();
        UpdateSummaryLabel();
    }
}

void OnFirstClassSelected(object sender, CheckedChangedEventArgs e)
{
    if (e.Value)
    {
        selectedClass = "Первый класс";
        SaveSelectedClass();
        UpdateSummaryLabel();
    }
}

async void OnConfirmSelection(object sender, System.EventArgs e)

```

```

    {
        // Возвращаем данные на предыдущую страницу
        MessagingCenter.Send(this, "UpdatePassengerClassLabel",
SummaryLabel.Text);
        await Navigation.PopModalAsync();
    }

    void SaveSelectedClass()
    {
        Application.Current.Properties["SelectedClass"] = selectedClass;
    }

    void UpdateSelectedClass()
    {
        switch (selectedClass)
        {
            case "Эконом":
                EconomyClassLabel.IsChecked = true;
                break;
            case "Комфорт":
                ComfortClassLabel.IsChecked = true;
                break;
            case "Бизнес":
                BusinessClassLabel.IsChecked = true;
                break;
            case "Первый класс":
                FirstClassLabel.IsChecked = true;
                break;
        }
    }

    void UpdateSummaryLabel()
    {
        int totalPassengers = int.Parse(AdultsLabel.Text) +
int.Parse(ChildrenLabel.Text) + int.Parse(InfantsLabel.Text);
        SummaryLabel.Text = $"{totalPassengers}, {selectedClass}";
    }
}
}

```

Page2.xaml:

```

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="App3.Page2"
    BackgroundColor="#242424">

    <Grid BackgroundColor="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*" />
            <RowDefinition Height="Auto" />
        </Grid.RowDefinitions>

        <!-- Размещаем содержимое страницы в верхней строке Grid -->
        <StackLayout Grid.Row="0" Padding="20">
            <!-- Логин пользователя и кнопка выхода -->
            <Label x:Name="usernameLabel" FontSize="20" TextColor="White"
HorizontalOptions="Center" Margin="0,10,0,10" IsVisible="False"/>
            <Button x:Name="logoutButton" Text="Выйти" BackgroundColor="#007AFF"
FontSize="18" TextColor="White" HorizontalOptions="Center" WidthRequest="100"
CornerRadius="12" TextTransform="None" Clicked="OnLogoutButtonClicked"
IsVisible="False"/>

            <!-- Войдите в профиль -->

```

```

        <Label x:Name="loginPromptLabel" Text="Войдите в профиль" FontSize="24"
        TextColor="White" HorizontalOptions="Center" Margin="0,10,0,30" TextTransform="None"
        FontFamily="Roboto-Medium" />
        <Button x:Name="loginButton" Text="Войти" BackgroundColor="#007AFF"
        FontSize="18" TextColor="White" HorizontalOptions="Center" WidthRequest="300"
        CornerRadius="12" TextTransform="None" FontFamily="Roboto-Medium"
        Clicked="OnLoginPage" />

        <!-- Настройки -->
        <Frame BackgroundColor="#323232" CornerRadius="10" Padding="10"
        Margin="0,60,0,0">
            <StackLayout>
                <Label Text="Настройки" FontSize="20" TextColor="White"
                Margin="0,0,0,20"/>
                <Label x:Name="RegionalSettingsLabel" Text="Региональные"
                FontSize="16" TextColor="White" Padding="0,10" />
                <BoxView HeightRequest="1" Color="#5E5E5E" />
                <Label x:Name="PriceDisplaySettingsLabel" Text="Отображение цен"
                FontSize="16" TextColor="White" Padding="0,10" />
                <BoxView HeightRequest="1" Color="#5E5E5E" />
                <Label x:Name="PrivacySettingsLabel" Text="Конфиденциальность"
                FontSize="16" TextColor="White" Padding="0,10" />
            </StackLayout>
        </Frame>
    </StackLayout>

    <!-- Размещаем кнопки в нижней строке Grid -->
    <StackLayout Grid.Row="2" Orientation="Horizontal" BackgroundColor="#323232"
    Padding="0" HeightRequest="60">
        <StackLayout Margin="35,5,20,5" VerticalOptions="Center">
            <ImageButton Source="Plane.png" BackgroundColor="Transparent"
            HeightRequest="30" WidthRequest="30" Clicked="OnMainButtonClicked" />
            <Label Text="Авиабилеты" TextColor="White"
            HorizontalTextAlignment="Center" FontSize="12" />
        </StackLayout>
        <StackLayout Margin="25,5,20,5" VerticalOptions="Center">
            <ImageButton Source="Calendar.png" BackgroundColor="Transparent"
            HeightRequest="30" WidthRequest="30" Clicked="OnFavorite" />
            <Label Text="Мои билеты" TextColor="White"
            HorizontalTextAlignment="Center" FontSize="12" />
        </StackLayout>
        <StackLayout Margin="30,5,5,5" VerticalOptions="Center">
            <ImageButton Source="Person.png" BackgroundColor="Transparent"
            HeightRequest="30" WidthRequest="30" />
            <Label Text="Профиль" TextColor="White"
            HorizontalTextAlignment="Center" FontSize="12" />
        </StackLayout>
    </StackLayout>
</Grid>
</ContentPage>

```

Page2.cs:

```

using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

```

namespace App3

```

{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class Page2 : ContentPage
    {
        public Page2()
        {
            InitializeComponent();

```

```

        NavigationPage.SetHasNavigationBar(this, false);
        InitializePage();
    }

    public Page2(string username)
    {
        InitializeComponent();
        NavigationPage.SetHasNavigationBar(this, false);
        InitializePage(username);
    }

    private void InitializePage(string username = null)
    {
        if (!string.IsNullOrEmpty(username))
        {
            usernameLabel.Text = $"Пользователь: {username}";
            usernameLabel.IsVisible = true;
            logoutButton.IsVisible = true;
            loginPromptLabel.IsVisible = false;
            loginButton.IsVisible = false;

            // Сохраняем информацию о пользователе
            Application.Current.Properties["username"] = username;
            Application.Current.SavePropertiesAsync();
        }
        else
        {
            usernameLabel.IsVisible = false;
            logoutButton.IsVisible = false;
            loginPromptLabel.IsVisible = true;
            loginButton.IsVisible = true;
        }

        // Добавляем обработчики жестов
        var regionalSettingsTap = new TapGestureRecognizer();
        regionalSettingsTap.Tapped += async (s, e) => {
            await Navigation.PushModalAsync(new RegionalSettingsPage());
        };
        RegionalSettingsLabel.GestureRecognizers.Add(regionalSettingsTap);

        var priceDisplaySettingsTap = new TapGestureRecognizer();
        priceDisplaySettingsTap.Tapped += async (s, e) => {
            await Navigation.PushModalAsync(new PriceDisplaySettingsPage());
        };

        PriceDisplaySettingsLabel.GestureRecognizers.Add(priceDisplaySettingsTap);

        var privacySettingsTap = new TapGestureRecognizer();
        privacySettingsTap.Tapped += async (s, e) => {
            await Navigation.PushModalAsync(new PrivacySettingsPage());
        };
        PrivacySettingsLabel.GestureRecognizers.Add(privacySettingsTap);
    }

    private async void OnLoginPage(object sender, EventArgs args)
    {
        Login logins = new Login();
        await Navigation.PushAsync(logins);
    }

    private async void OnMainButtonClicked(object sender, EventArgs e)
    {
        MainPag MainPage = new MainPag();
        await Navigation.PushAsync(MainPage);
    }

```

```

    }

    private async void OnLogoutButtonClicked(object sender, EventArgs e)
    {
        bool confirm = await DisplayAlert("Подтверждение", "Вы действительно  
хотите выйти?", "Да", "Нет");
        if (confirm)
        {
            // Удаляем информацию о пользователе
            Application.Current.Properties.Remove("username");
            await Application.Current.SavePropertiesAsync();

            // Скрываем информацию о пользователе и показываем кнопки входа
            usernameLabel.IsVisible = false;
            logoutButton.IsVisible = false;
            loginPromptLabel.IsVisible = true;
            loginButton.IsVisible = true;
        }
    }

    private async void OnFavorite(object sender, EventArgs e)
    {
        // Создание новой страницы профиля
        var favoritesPage = new FavoriteFlightsPage();
        await Navigation.PushAsync(favoritesPage);
    }
}

```

Regist.Page.xaml:

```

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="App3.RegisterPage"
    BackgroundColor="#2B2B2B">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*" />
            <RowDefinition Height="Auto" />
        </Grid.RowDefinitions>

        <StackLayout Padding="30" VerticalOptions="CenterAndExpand" Grid.Row="0">
            <Label Text="Регистрация" TextColor="White" FontSize="Large"
                HorizontalOptions="Center" Margin="0,20,0,20"/>
            <Frame BackgroundColor="#333" CornerRadius="10" Padding="10">
                <StackLayout>
                    <Entry x:Name="usernameEntry" Placeholder="Логин (Email)"
                        TextColor="White" PlaceholderColor="Gray"/>
                    <Label x:Name="usernameErrorLabel" Text="Неверный формат email"
                        TextColor="Red" IsVisible="False"/>
                    <Entry x:Name="passwordEntry" Placeholder="Пароль"
                        IsPassword="True" TextColor="White" PlaceholderColor="Gray"/>
                    <Entry x:Name="confirmPasswordEntry" Placeholder="Подтвердите
                        пароль" IsPassword="True" TextColor="White" PlaceholderColor="Gray"/>
                    <StackLayout Orientation="Horizontal">
                        <CheckBox x:Name="showPasswordCheckBox"
                            CheckedChanged="OnShowPasswordCheckBoxChanged"/>
                        <Label Text="Показать пароль" TextColor="White"
                            VerticalOptions="Center"/>
                    </StackLayout>
                </StackLayout>
            </Frame>
            <Button Text="Регистрация" TextColor="White" BackgroundColor="#1E88E5"
                CornerRadius="5" Margin="0,20,0,0" Clicked="OnRegisterButtonClicked"/>
        </ContentPage>

```

```

        </StackLayout>
    </Grid>
</ContentPage>

```

Regist.Page.cs:

```

using System;
using System.Security.Cryptography;
using System.Text;
using System.Text.RegularExpressions;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace App3
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class RegisterPage : ContentPage
    {
        public RegisterPage()
        {
            NavigationPage.SetHasNavigationBar(this, false);
            InitializeComponent();
            passwordEntry.IsPassword = true;
            confirmPasswordEntry.IsPassword = true;
        }

        private async void OnRegisterButtonClicked(object sender, EventArgs e)
        {
            string username = usernameEntry.Text;
            string password = passwordEntry.Text;
            string confirmPassword = confirmPasswordEntry.Text;

            if (!IsValidEmail(username))
            {
                usernameErrorLabel.IsVisible = true;
                return;
            }
            else
            {
                usernameErrorLabel.IsVisible = false;
            }

            if (password != confirmPassword)
            {
                await DisplayAlert("Ошибка", "Пароли не совпадают", "OK");
                return;
            }

            string hashedPassword = HashPassword(password);

            User newUser = new User
            {
                Username = username,
                PasswordHash = hashedPassword
            };

            App.Db.SaveUser(newUser);

            await DisplayAlert("Успех", "Регистрация прошла успешно", "OK");
            await Navigation.PopAsync();
        }

        private bool IsValidEmail(string email)
        {
            string emailPattern = @"^[^@\s]+@[^@\s]+\.[^@\s]+$";

```

```

        return Regex.IsMatch(email, emailPattern);
    }

    private string HashPassword(string password)
    {
        using (var sha256 = SHA256.Create())
        {
            byte[] bytes = sha256.ComputeHash(Encoding.UTF8.GetBytes(password));
            StringBuilder builder = new StringBuilder();
            for (int i = 0; i < bytes.Length; i++)
            {
                builder.Append(bytes[i].ToString("x2"));
            }
            return builder.ToString();
        }
    }

    private void OnShowPasswordCheckBoxChanged(object sender,
CheckedChangedEventArgs e)
    {
        passwordEntry.IsPassword = !e.Value;
        confirmPasswordEntry.IsPassword = !e.Value;
    }
}


```

Приложение Б

Формы входных и выходных документов

Android Emulator - pixel_5_-_api_33:5554

00:17



Аэропорт вылета: Аэропорт Лос-Анджелес (США)

Аэропорт прилета: Аэропорт Сан-Франциско (США)

Номер рейса: UA456

Класс обслуживания: Эконом

Дата и время вылета: 02.07.2024 23:29:09

Дата и время прилета: 03.07.2024 5:29:09

Время полета: 1,5 часов

Цена: 2400 Bp

Количество мест: 220

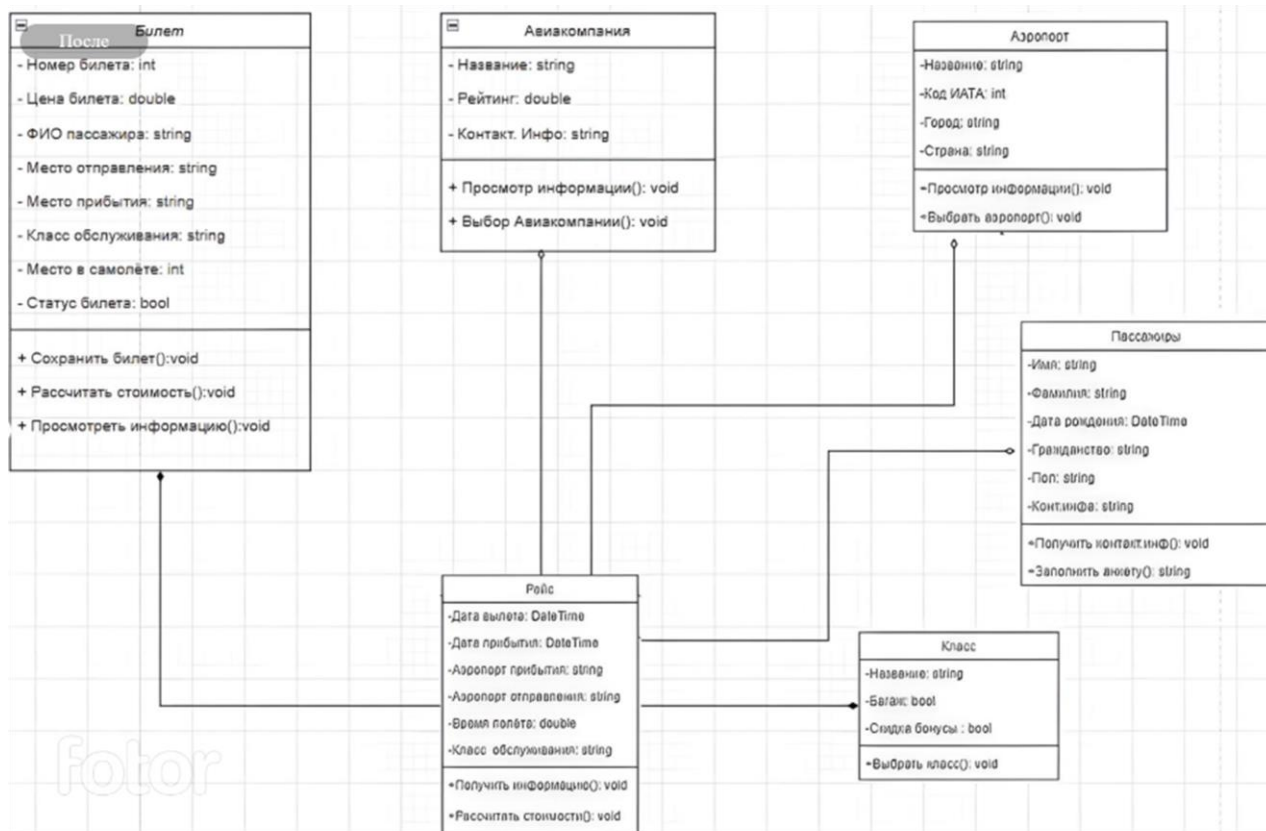
Описание страны: США - страна разнообразия и возможностей.

СОХРАНИТЬ В ИЗБРАННЫЕ

СОХРАНИТЬ КАК PDF

ЗАКРЫТЬ

Рисунок Б.1 – Выгрузка выбранного рейса



Разработка мобильного приложения «Калькулятор расчёта стоимости авиабилетов авиакомпании «Белавиа»

Диаграмма классов

Лит.	Масса	Масштаб
У		
Лист 1	Листов 4	82

КБП

Подп. и дата

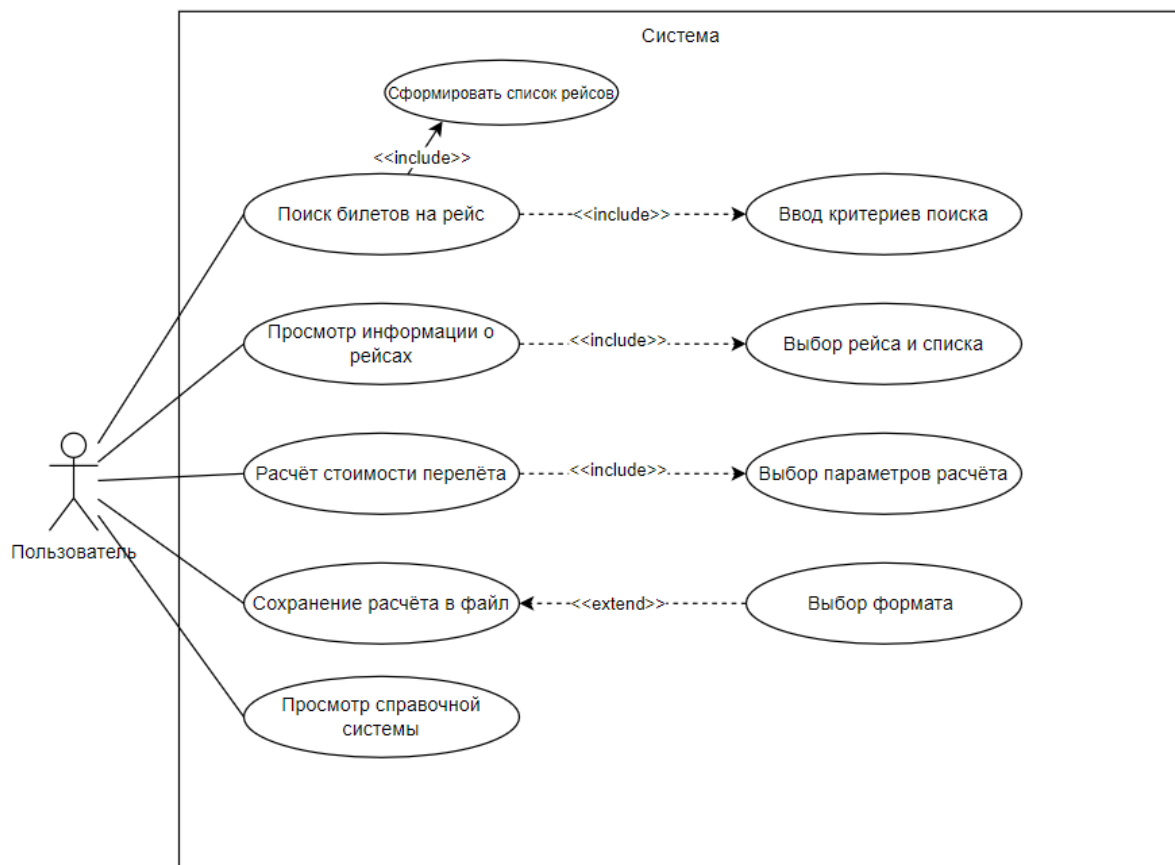
Инв.№дубл.

Взам.инв.№

Подп. и дата

Инв.№подл.

Изм.	Лист	№ докум.	Подпись	Дата
Разраб.		Носко А.А.		
Провер.		Коропа Е.Н.		
Н. Контр.				
Реценз.				
Т. Контр.				
Утверд.				



Подп. и дата

Имя, Подпись

Взам. инв. №

Подп. и дата

Имя, Подпись

КП Т.192010.401 ГЧ

Разработка мобильного приложения «Калькулятор расчёта стоимости авиабилетов авиакомпании «Белавиа»

Диаграмма вариантов использования

Лит.

Масса

Масштаб

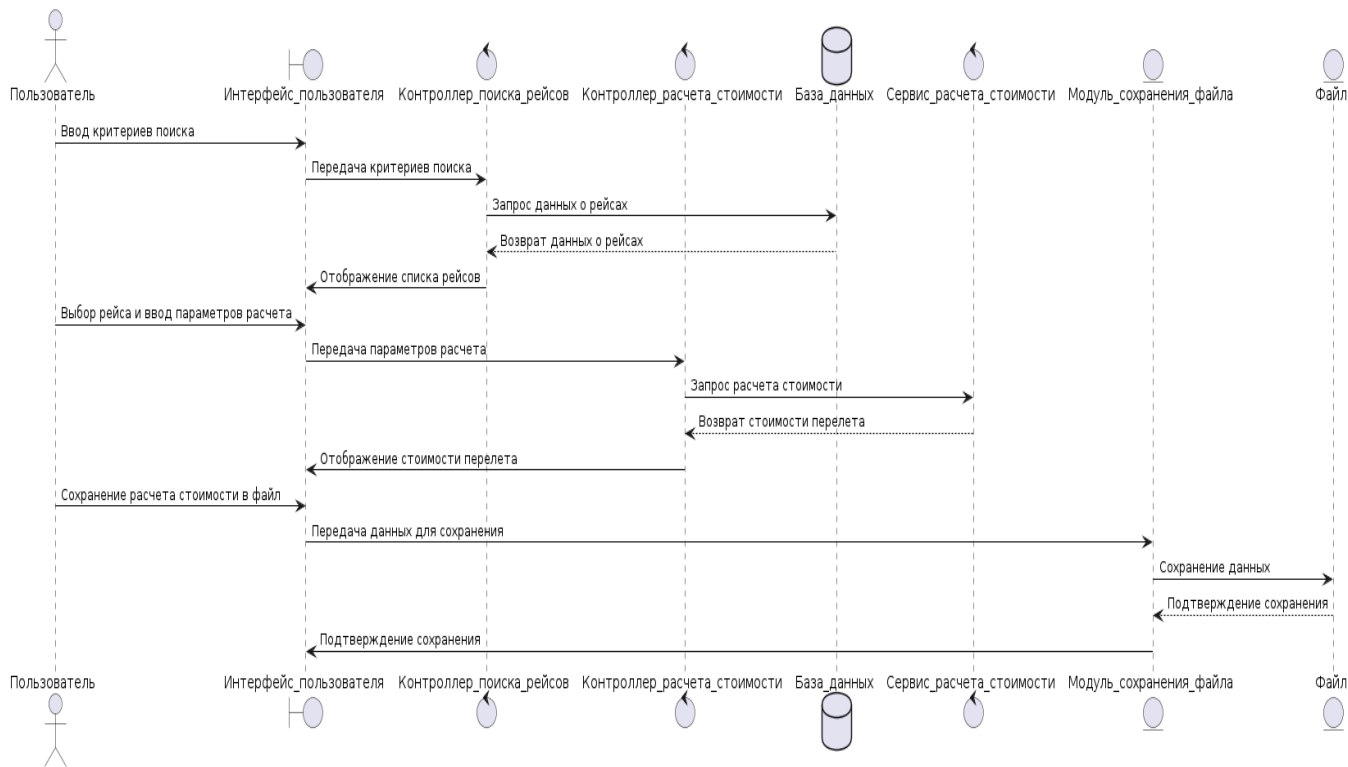
У

Лист 1

Листов 4 83

КБП

Изм.	Лист	№ докум.	Подпись	Дата
Разраб.		Носко А.А.		
Провер.		Коропа Е.Н.		
Н. Контр.				
Реценз.				
Т. Контр.				
Утверд.				



Подп. и дата

Инв.№дубл.

Взам.инв.№

Подп. и дата

Инв.№подл.

КП Т.192010.401 ГЧ

Разработка мобильного приложения «Калькулятор расчёта стоимости авиабилетов авиакомпании «Белавиа»

Лит.

Масса

Масштаб

у

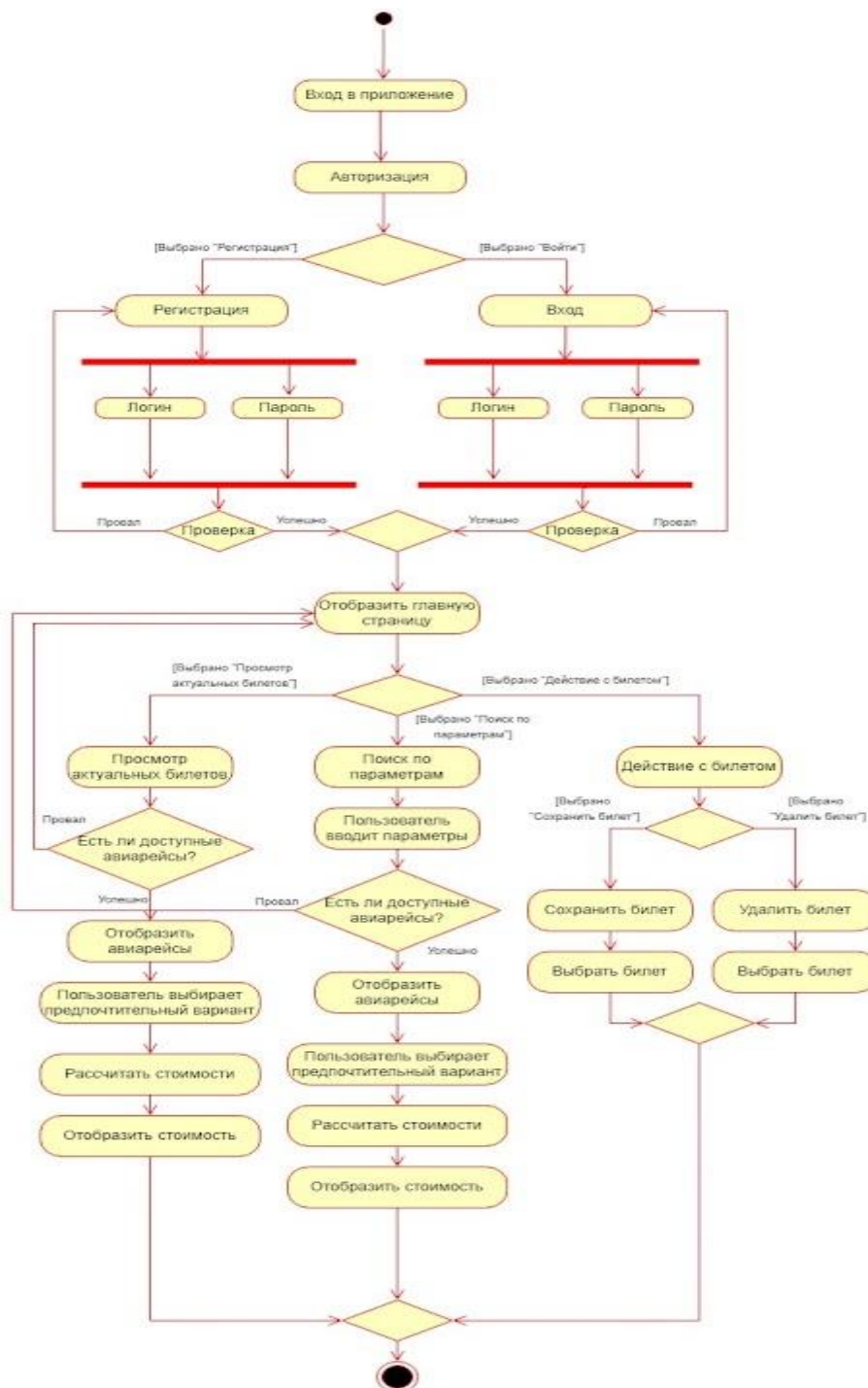
Лист 1

Листов 4 84

Диаграмма последовательности

КБП

Изм.	Лист	№ докум.	Подпись	Дата
Разраб.		Носко А.А.		
Провер.		Корона Е.Н.		
Н. Контр.				
Реценз.				
Т. Контр.				
Утверд.				



КП Т.192010.401 ГЧ

Разработка мобильного приложения «Калькулятор расчёта стоимости авиабилетов авиакомпании «Белавиа»

Диаграмма деятельности

Лит.	Масса	Масштаб
У		
Лист 1	Листов 4	85

КБП

Изм.	Лист	№ докум.	Подпись	Дата
Разраб.		Носко А.А.		
Провер.		Корона Е.Н.		
Н. Контр.				
Реценз.				
Т. Контр.				
Утверд.				

Подп. и дата

Инв. №докум.

Взам. инв. №

Подп. и дата

Инв. №подл.

Этикетка
для курсовых проектов

Курсовой проект

Тема Разработка мобильного приложения «Калькулятор расчёта стоимости авиабилетов авиакомпании «Белавиа»

КП Т.192010.401

Разработан _____

Утвержден 15.02.2024

Руководитель: Коропа Е.Н.

Технические средства: Технические средства: смартфоны (Android 11 и выше), планшеты (Android 11 и выше)

Программные средства: NETStandard.Library (2.0.3), sqlite-net-pcl (1.9.172), Xamarin.Essentials (1.8.1), Xamarin.Forms (5.0.0.2662)

Состав документа:

Пояснительная записка – Носко ПЗ.docx

Графическая часть – ДВИ.docx, Последовательности.docx, Классов.docx, Деятельности.docx

Папка с проектом – App3

Установочный пакет программного средства – com.companyname.app3.apk

Файлы базы данных – bd.sqlite3

Сведения о защите информации: пользователь: логин 1@mail.ru, пароль 1

Удостоверяющий лист
электронного документа – курсовой проект

Тема Разработка мобильного приложения «Калькулятор
расчёта стоимости авиабилетов авиакомпании «Белавиа»

Обозначение КП Т.192010.401

Разработчик Носко А.А. Руководитель Коропа Е.Н.
(Ф.И.О.) (Ф.И.О.)

Подписи лиц, ответственных за разработку электронного документа

Состав электронного документа	Разработчик	Руководитель
Пояснительная записка (на бумажном носителе формата А4), Носко ПЗ.doc		
ГЧ, ДВИ.docx		
ГЧ, Последовательности.docx		
ГЧ, Классов.docx		
ГЧ, Деятельности.docx		
Папка с проектом «App3»		
Установочный пакет программного средства «com.compraname.app3.apk»		
Тип носителя: флеш-накопитель		